An adaptive spherical view representation for mobile robot navigation in non-stationary environments

Feras Dayoub

A thesis submitted in partial fulfilment of the requirements of the
University of Lincoln for the degree of
Doctor of Philosophy

December 2011

Abstract

One of the major goals of mobile robotics is to introduce functional and useful mobile service robots to human environments, which means that these robots have to be able to build and maintain maps of a dynamic and ever-changing world. This includes adapting to changes in the appearance of the environment – changes that may be spontaneous, discontinuous and unpredictable – as a result of human activities. However, most past research on robot mapping addresses only the initial learning of an environment, a phase which will only be a short moment in the lifetime of a service robot that may be expected to operate for many years.

In order to maintain an up-to-date map of a non-stationary environment over a lifetime, the robot can use its continuous stream of sensory information, which reflects the momentary status of its surroundings, to update the map. However, the amount of sensory information to be processed in a lifetime is vast; therefore, efficient methods are required for filtering, storing and updating this information over time. To this end, this thesis presents a long-term map-updating mechanism inspired by the multi-store model of human memory. The updating mechanism is integrated with a hybrid visual map that represents the global topology and local geometry of a non-stationary environment. The map consists of an adjacency graph of nodes on a global level, and each node on the local level of the map represents a spherical view representation of image features extracted from an omnidirectional image of the node. The spherical views provide both an appearance signature for the nodes, which the robot uses to localise itself in the

environment, and heading information when the robot uses the map for visual navigation.

The system is evaluated using a series of extensive experimental analyses that demonstrate the persistence performance of the system and its key components, involving long-term experiments in localisation, heading estimation and navigation in real non-stationary indoor office environments over long periods of time.

Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation

Mobile service robots, robotic helpers and companions have become a major topic for current research in robotics. This trend has been facilitated by the increased performance of modern processors and the emergence of low-cost, high performance sensors such as digital cameras, together with recent advances in academic disciplines including computer vision, artificial intelligence and probabilistic robotics. Simple robots such as vacuum cleaners and lawn mowers are already available to consumers, while Microsoft's Bill Gates predicted that personal service robots will become common in the home and office of the future in the same way as the PCs and mobile phones of today (Gates, 2007).

Common to all such robots is that they will share physical spaces with humans, and will thus need to deal with a dynamic and ever-changing world. This includes adapting to changes in the arrangement of objects and the appearance of the environment – changes that may be spontaneous, discontinuous and un-

predictable – as a result of human activities.

For example, the appearance of a room in a house is not static over time: new objects are sometimes added, existing objects such as pictures or carpets may be changed from time to time and old objects may be removed; other changes may occur at much faster speed, such as moving chairs and cushions. For any mobile robot which needs to work among humans in such environments for a long period of time, having the ability to detect these changes would enable the robot to maintain an up-to-date internal spatial representation (map) of its environment. This is important because allowing a map to become out-of-date can result in problems when performing tasks which require the use of the map, such as self-localisation (i.e. answering the question "Where am I?") and path planning (i.e. answering the question "How to get from A to B?"). These problems arise because the information stored in the map will no longer reflect the true status of the robot's environment.

1.2 The Problem

The main challenge in mapping non-stationary environments by mobile robots comes from the fact that the appearance and the configuration of the environment can change in unpredictable ways.

A naive solution to this problem would be simply to re-build the map from time to time. However, this would lead to a discontinuity in the work of the robot due to the need of repeatedly building a new map. In addition, some other agent, e.g. a human supervisor, would be needed to decide whether the robot's map needed to be rebuilt. Instead, this thesis proposes an automatic mechanism

for detecting changes in the environment and updating the robot's map to more accurately reflect the true state of the environment.

In this thesis, we argue that the solution to the problem of mapping non-stationary environments for indefinitely long periods of time should meet the following requirements:

- **Accuracy.** The most obvious requirement is to represent faithfully the true state of the environment. In a dynamic environment, the map should accurately reflect changes, while the accuracy of the map should increase with more sensor measurements even when the environment remains static.

- **Computational requirements.** While it may be possible to maintain an off-line database of all past observations of an environment, on-line operation requires a compact representation that remains bounded during long-term operation, since a service robot must operate in real-time with limited computational resources. Maintaining compactness in the robot's map requires a balance between learning and forgetting.

- **Stability.** A key issue for any long-term mapping and localisation system is the danger that the interdependency between mapping and localisation introduces a positive feedback loop, where measurement noise may be picked up and amplified to the point where the system becomes unstable (see related discussion in (Ess et al., 2008)). An important question is therefore how to avoid such instabilities and maintain robust, long-term performance.

- **Delayed declaration of outliers.** The fact that the robot is required to work for long periods of time using its map, which needs to be updated

over time, adds a fourth dimension (time) to the problem and introduces further complexity. In particular, actual changes in the environment appear in the first instance as outliers, which can only be identified after more time has passed and more measurements have been made (a problem identified previously in (Biber and Duckett, 2005)). Therefore, the map representation should be able to track multiple hypotheses until it can be determined whether or not a change has really happened.

- **Real-time versus off-line processing.** During long-term operation, the robot has to perform some tasks under hard real-time constraints (e.g. self-localisation), while other tasks may be performed in the background when reasonable delays are acceptable, such as while charging or between service tasks.

Following the above requirements, the work in this thesis introduces a systematic mechanism to adapt a robot's map over time by incrementally learning the new features of a changing environment and, at the same time, forgetting old and obsolete information. A discussion about how our system meets the above requirements is presented in the conclusion of this thesis, in Chapter 7.

1.3 The Proposed Solution

The proposed solution presented in this work assumes that a mobile service robot is required to work in a human-centred indoor environment for a long period of time. The workplace of the robot is subject to frequent changes as a result of human activities.

Similar to the situation when a new member of staff is introduced to a workplace, the robot builds an initial map from its first tour of the environment. After this initial step, the robot continually updates the information stored in the map in response to any changes in the appearance of the environment, which is achieved by a long-term updating mechanism inspired by the multi-store model of human memory (Atkinson and Shiffrin, 1968).

In this thesis it is assumed that the initial topology in the robot's map remains fixed over time, whereas the visual information stored locally in the nodes of the map is subject to changes. This assumption is based on the observation that indoor office-type environments typically have an underlying structure, containing fixed walls, doorways, and corridors, which means that human activities are usually constrained to altering the appearance of the environment but not its general topology. Of course, this assumption will be broken occasionally due to changes in the structure of a building, for example the re-arrangement of office dividers in an open-plan office or the location of shelves in a supermarket, and in extreme cases removing or adding walls between two rooms. Chapter 7 provides a discussion on possible solutions to address this limitation in future work.

The following sections give a brief overview of the spatial representation of the environment used in this thesis and the long-term updating mechanism.

1.3.1 Spatial Representation

The spatial representation of the environment used in this thesis is a hybrid map, which presents the environment as a combination of two or more levels of distinct types of map (Buschka and Saffiotti, 2004). The map consists of two levels, global

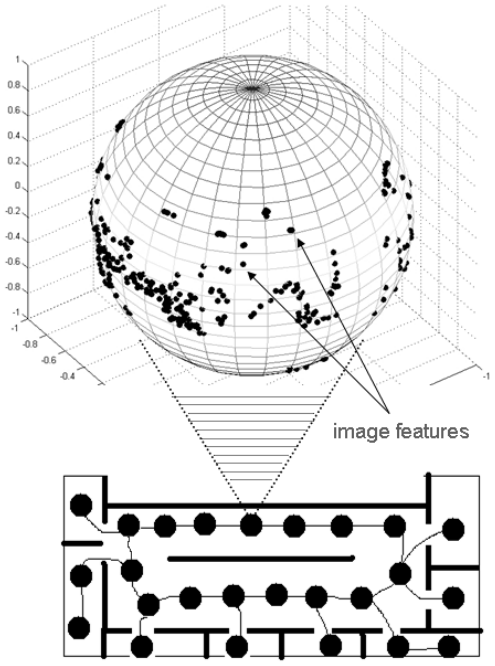

Figure 1.1: Proposed hybrid map with two levels, global and local. The environment is represented as an adjacency graph of nodes on the global level of the map, and each node on the local level represents a spherical view of local image features.

and local. On the global level, the world is represented as a sparse topological map labelled with geometric information concerning the relative pose (position and orientation) of its nodes, i.e. an adjacency graph where the nodes of the graph correspond to certain places in the environment. On the local level, each node is represented as a local map of stored image features. The local maps are based on a spherical view representation, containing a stored set of local image features mapped onto the surface of a unit sphere (see Fig. 1.1).

The robot uses the global part of the map to localise itself in one of the nodes (comparing the robot's current view to all of the stored views in the map), and then uses the information stored in the local map corresponding to the current

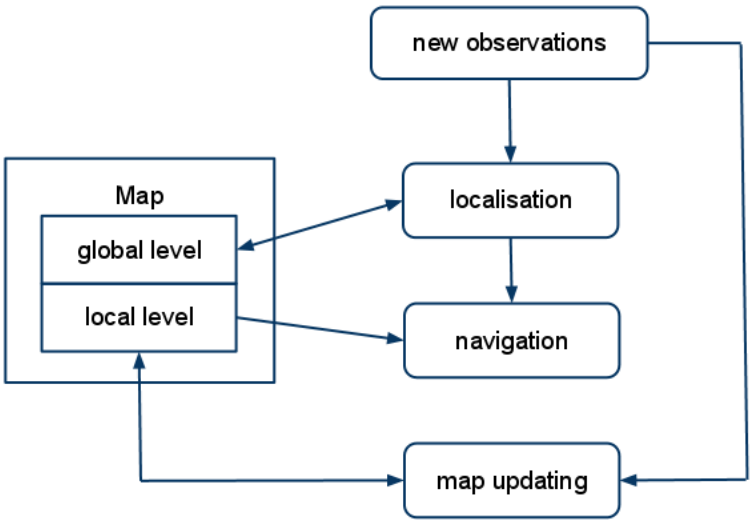

Figure 1.2: Overall view of the system. The robot's internal representation of the world is a map consisting of two levels, global and local. The robot uses its sensory observations to localise itself in the global level of the map. Then it uses the same observations to estimate its heading using the view representation stored in the local map corresponding to the current node. In turn, observed changes in the environment are used to update the map.

node to estimate its heading for navigating through the environment. The heading estimation is achieved using the 3D bearing of the stored image features in the spherical view representation, which is calculated by applying epipolar geometry for spherical cameras (Akihiko and Atsushi, 2005).

The nodes on the global level of the map are selected automatically using a dual clustering algorithm, which employs two properties to select the nodes, namely the metric distance between the nodes and the similarity between the image features stored in these nodes. The algorithm selects a minimal set of nodes to cover the entire environment, while avoiding any "gaps" in the map which could lead to navigation failures. The mapping method is explained in

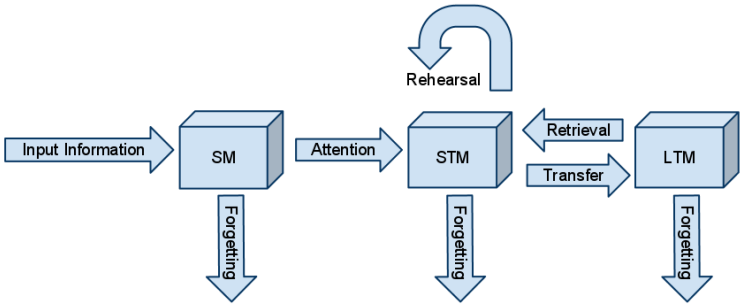

Figure 1.3: Multi-store memory model. SM: Sensory memory. STM: Short-term memory. LTM: Long-term memory. Selective attention, which involves the LTM, determines what information moves from SM to STM. Through the process of rehearsal, information in STM can be transferred to LTM and be retained for longer periods of time. Information from the LTM store is retrieved using a process called "recall".

more detail in Chapter 4. Fig. 1.2 gives an overview of the whole system presented in this thesis.

1.3.2 Multi-Store Memory Model

To manage the time-related aspects of the system, an information processing model based on the multi-store model of human memory (Atkinson and Shiffrin, 1968) is introduced, which divides memory into three stores: sensory memory (SM), short-term memory (STM) and long-term memory (LTM) (see Fig. 1.3).

In the multi-store model of the human memory, the sensory memory contains information perceived by the senses, and selective attention determines what information is moved from sensory memory to STM. Through the process of rehearsal, information in STM can be committed to LTM to be retained for longer periods of time. In turn, the knowledge stored in LTM affects perception and is used by the attentional mechanism to select which information to attend

to in the environment. This model forms the basis of nearly all modern theories of human memory. Note that there is no attempt in this thesis to emulate the underlying biological mechanisms (e.g. hippocampal neurons): this model serves only as a high-level abstraction for temporal information processing.

Applying these concepts to the robot's hybrid map, separate STM and LTM stores are maintained for each node in the map. STM is used as a temporary store for new measurements and LTM contains stored feature data (comprising local image features and their estimated location in the spherical view representation), which are used for global localisation and heading estimation. More details about the memory model and the processes for updating the STM and LTM stores are presented in Chapter 5.

In order to test the model, several experiments were conducted using data collected from real changing environments over long periods of time, which are reported in Chapter 6.

1.4 Contributions

This thesis presents a system that enables a mobile robot to maintain an up-to-date internal representation of the appearance of its environment. The environment is non-stationary and the robot is required to operate inside it for a long period of time. This internal representation is also used by the robot to perform tasks such as visual navigation and self-localisation. The specific contributions presented in this thesis include:

- A computational model based on the multi-store model of human memory and used for updating a set of stored features (landmarks) for a particu-

lar place in the environment, and its application by mobile robots to the problem of global localisation in non-stationary environments.

- An adaptive spherical view representation for estimating the 3D bearing of landmarks for a particular place in the environment, based on the above memory model, and its application by mobile robots to the problem of heading estimation and navigation in non-stationary environments.
- A dual clustering technique used to create a sparse initial topological map of an indoor environment.
- The integration of the above techniques into a complete working system for long-term navigation by a mobile service robot, based on a fixed topological map with continual adaptation of the stored view representations in the map.
- An extensive experimental analysis of the system and its key components, involving long-term experiments in localisation, heading estimation and navigation in indoor office environments, including analysis of the long-term stability of the approach.

1.5 Publications

Part of the content of this thesis has already been presented in a number of journal articles, conferences and workshops. Here is a complete list of publications produced during the course of this Ph.D. study.

Journal Article

- Feras Dayoub, Grzegorz Cielniak and Tom Duckett. Long-Term Experiments with an Adaptive Spherical View Representation for Navigation in Changing Environments. Robotics and Autonomous Systems, Volume 59, Issue 5, pp. 285-295, May 2011.

Conference Proceedings

- Feras Dayoub, Grzegorz Cielniak and Tom Duckett. A Sparse Hybrid Map for Vision-Guided Mobile Robots. In Proceedings of the 5th European Conference on Mobile Robots (ECMR), Örebro, Sweden, pp. 213-218, 7-9 September 2011.

- Feras Dayoub, Grzegorz Cielniak and Tom Duckett. Long-Term Experiment Using an Adaptive Appearance-Based Map for Visual Navigation by Mobile Robots. Extended Abstract. In Proceedings of the 12th Conference Towards Autonomous Robotic Systems (TAROS), LNAI 6856, pp. 400–401. Springer, Heidelberg (2011).

- Feras Dayoub, Tom Duckett, and Grzegorz Cielniak. An Adaptive Spherical View Representation for Navigation in Changing Environments. In Proceedings of the 4th European Conference on Mobile Robots (ECMR), Dubrovnik, Croatia, pp. 1-6, 23-25 September 2009.

- Feras Dayoub and Tom Duckett. An Adaptive Appearance-Based Map for Long-Term Topological Localization of Mobile Robots. In Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS), Nice, France, pp. 3364-3369, 22-26 September 2008.

Workshop and Symposium Papers

- Feras Dayoub, Tom Duckett and Grzegorz Cielniak. Toward an Object-Based Semantic Memory for Long-Term Operation of Mobile Service Robots. In Workshop on Semantic Mapping and Autonomous Knowledge Acquisition, International Conference on Intelligent Robots and Systems (IROS), Taipei, Taiwan, 18-22 October 2010.

- Feras Dayoub, Tom Duckett and Grzegorz Cielniak. Short-and Long-Term Adaptation of Visual Place Memories for Mobile Robots. In Workshop on Remembering Who We Are – Human Memory for Artificial Agents Symposium, Artificial Intelligence and Simulation of Behaviour (AISB), Leicester, UK, 29th March - 1st April 2010.

1.6 Outline of the Thesis

The remainder of this thesis is organised as follows:

- **Chapter 2** reviews the state-of-art and related work in the mapping of stationary and non-stationary environments for mobile robots.

- **Chapter 3** introduces the essential components of the proposed system including the image features used in this work and the set-up of the omni-directional imaging system and its related multi-view geometry.

- **Chapter 4** presents the method used to create the initial hybrid map of a mobile robot's environment. The chapter also proposes a novel method for automatically selecting the nodes of the map based on a dual clustering approach.

- **Chapter 5** describes a method for updating the spherical view representations of the map over time. The updating method is inspired by the multi-store model of human memory.
- **Chapter 6** contains details of the experiments which were conducted to evaluate the system.
- **Chapter 7** concludes the thesis, presenting open questions, limitations of the proposed system and possible topics for future work.

Chapter 2

Literature Review

This chapter reviews related work on mobile robot mapping. The chapter categorises mapping methods into two groups based on the assumed dynamic nature of the robot's environment. The two groups contain the methods which assume a static and non-static environment, respectively.

2.1 Introduction

Mobile robots need to know where they are in the environment (localise themselves) in order to perform many useful tasks, such as path planning and navigation. Localisation can be achieved if a map of the environment is already provided to the robot. However, such maps are not always available, which means that the robot needs to build the map using sensory information while moving through the environment. In other words, the robot needs to build the map while simultaneously localising itself relative to that map. The problem is referred to as "simultaneous localisation and mapping" (SLAM).

Due to the importance of solving the mapping and localisation problem before truly autonomous mobile robots can be built, the robotics community has given much attention to SLAM over the past decade. The result is a wide variety of mapping methods with their own advantages and disadvantages (Thrun and Leonard, 2008). The criteria against which these mapping methods are generally evaluated involve the quality of the map they produce and the scale and complexity of the environment which they are able to handle.

The mapping process is generally carried out either on-line, by driving the robot manually, or autonomously, inside the environment while running the SLAM algorithm. Alternatively, the mapping process can be done off-line by using pre-collected data from a robot which has been driven through the environment. In both of these cases there are now a collection of robust SLAM algorithms available to be used and able to build accurate and high quality maps. For example, (Eliazar and Parr, 2003; Frese and Schroder, 2006; Grisetti et al., 2009; Kaess et al., 2008).

This thesis does not address the classic mapping problem where a mobile robot explores the environment and produces a map at the end, but instead focuses on the period after the mapping process is finished. Generally, in this period, the robot uses the resulting map to perform tasks such as autonomous navigation and path planning, assuming that the environment is static and the information in the map always reflects the true status of the world. However, this assumption cannot hold for real-life environments where the surroundings of the robot are ever-changing. Therefore, existing mapping methods need to be extended to accommodate the non-stationary nature of real-world environments. The result of this extension is a group of approaches that try to make the map

suitable for use in changing environments.

In the following sections, the mapping methods are divided into two groups based on the assumed dynamic nature of the robot's environment. One group of methods assumes that the environment is static and the other assumes a non-static environment. However, due to the large number of proposed mapping methods and the rich variety of approaches, it is difficult to divide all the methods into only two distinct groups without intersections regarding the way the methods represent the environment and the underling estimation framework which they use to build the map.

2.2 Mapping in Static Environments

Those methods which assume that the robot's environment is static are dominant within the literature. One way to classify them is based on the way they represent the environment, which in turn generates the following three classes of representation.

The first class includes metric methods, e.g. (Elinas et al., 2006; Se et al., 2002), which aim to generate a geometric map of the environment where the positions of landmarks are stored in the map relative to a global coordinate system. The second class includes topological methods, e.g. (Valgren et al., 2006; Zivkovic et al., 2007), which represent the environment as a graph, where the nodes of this graph correspond to places in the real environment. The third class includes hybrid methods, which represent the environment as a combination of two or more distinct maps of different types. The most common type of hybrid map combines topological and metric maps, e.g. (Kuipers et al., 2004).

Different sensors are used by these methods to build the map, but the two major types are laser range finders and cameras. While laser range finders can provide high resolution scans, which make it possible to reliably detect and localise corners and other edge features in the environment, they suffer from many problems such as high cost, high power demand and safety constraints. Cameras, on the other hand, are passive, low weight and less expensive with a high perceptual resolution, which makes them suitable as external sensors for mobile robots. To this end, this work uses an omnidirectional camera as the main sensor for the robot. Chapter 3 contains specific details about the computer vision system used in this thesis.

In the following sections, a short review for the three different mapping methods mentioned above is presented, including examples of the methods which use vision as the main sensor on board the robot.

2.2.1 Metric Methods

Metric mapping methods aim to build a map of the environment that represents the world using metric information, which underlines the relation between a group of landmarks extracted from the sensory data of the robot and their position relative to the origin of a global reference frame. The maps produced by these methods can be divided into two main types, namely occupancy grid maps, e.g. see Fig. 2.1, which represent the world as a grid of occupied and non-occupied cells (Elfes, 1989), and feature-based maps, e.g. see Fig. 2.2, which contain the locations of distinct features in the environment (Se et al., 2002).

Metric methods are the most popular and widespread approach for mobile

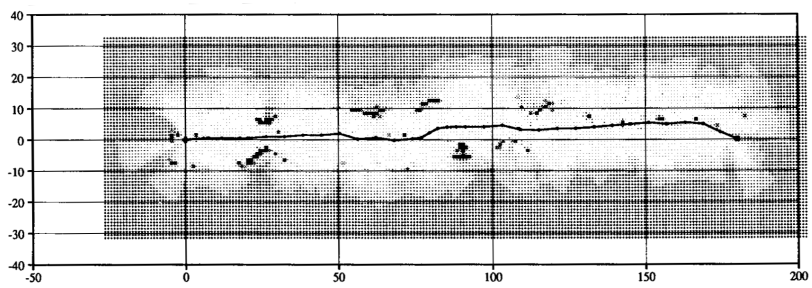

Figure 2.1: Sonar-based Occupancy Grid Map (Elfes, 1989).

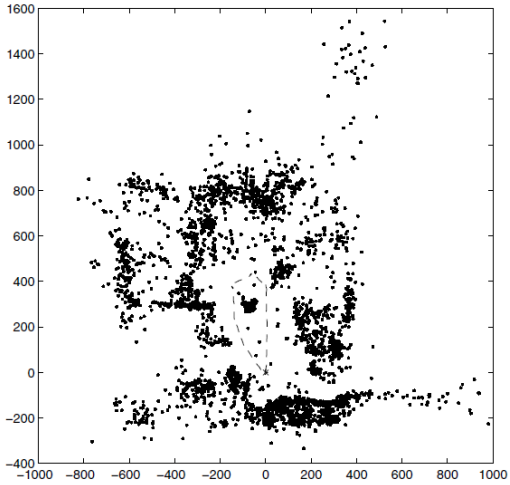

Figure 2.2: Feature-based Map (Se et al., 2002).

robot mapping. In general, they can be classified into three major paradigms based on their underlying estimation techniques (Thrun and Leonard, 2008), i.e. extended Kalman filter, particle filter and graph optimisation techniques.

2.2.1.1 Extended Kalman Filter (EKF)

This approach to metric mapping estimation was first introduced by Smith et al. (1987). The general technique uses a single state vector to represent the estimated robot and landmark's positions in the environment. Errors in the estimation

process are handled by a covariance matrix, which is updated incrementally as new observations become available from the sensors on board the robot (Newman, 1999).

One of the weak points of EKF-based methods is scalability because maintaining a single covariance matrix for all the landmarks in the map is computationally expensive when dealing with large-scale environments. A number of researchers have proposed algorithms to overcome the problem of scalability, which divide the global map into smaller sub-maps and maintain a separate covariance matrix for each one (Huang et al., 2006; Tardos et al., 2002). Another weak point of EKF-based methods is the inability to track multiple distinct hypotheses. This situation arises when ambiguity in the sensory information produces multiple hypotheses about the true location of the robot in the environment.

In the domain of vision-based EKF methods, the work presented by Davison et al. (2007) is one of the most successful examples of the real-time implementation of a mapping system under an EKF framework. The authors introduced a mapping system called MonoSLAM, which uses a hand-held camera and builds a map of the 3D positions of image features. The system state vector of the filter contains the 3D positions of all the image features in the map and the last pose of the camera. However, the need to maintain all positions of all the image features in one state vector limits the size of the environment in which the system can map with a real-time computation performance. In order to overcome this limitation, Clemente et al. (2007) split the filter of the MonoSLAM system into sub-maps, which extended the size of the mapped area to a 250 m loop trajectory.

2.2.1.2 Particle Filtering

These methods represent the uncertainty of the robot position and the map landmarks using a set of samples, or particles. Each particle represents a hypothesis of what the true position of the robot and the map may be. At each time step, particles are moved according to a motion model of the robot and each particle's map is then updated based on sensor observations. The particles are weighted according to the likelihood of the observations given by the sampled poses and previous observations. One of the most successful implementations of this type of method is FastSLAM, introduced by Montemerlo et al. (2002), which is based on a Rao-Blackwellised particle filter (Doucet et al., 2000) and uses a set of Kalman filters to represent the map features conditioned on a sampled robot trajectory. That is, each particle represents a guess at what the true full path of the robot may be, not just its last position. The particles are moved based on the odometry motion model and weighted by the likelihood of the observations.

In the field of computer vision-based methods, particle filtering techniques have shown good results. For example, Sim et al. (2005) used the 3D position of image features as landmarks in the map. The mapping system was based on the FastSLAM algorithm by which both the map and the trajectory of the robot were estimated, and the particles were moved based on the robot's motion, which was estimated in turn using visual stereo ego-motion (Se et al., 2001). Estimating the motion of the robot from vision instead of odometry simplifies the mapping problem due to less noisy and more accurate motion estimation.

2.2.1.3 Graph-based Optimisation

The output of these methods is a pose-graph, where the nodes associate past poses of the robot with map landmarks. The edges of the graph model spatial constraints between the nodes (Lu and Milios, 1997). The optimisation step aims to select the spatial configuration of the nodes, which best satisfies the constraints encoded in the edges. The result is the robot's best estimate for its map and its full trajectory.

Graph-based optimisation methods deal with the SLAM problem as a non-linear optimisation problem. Non-linearity comes from the fact that constraints in the graph include the rotational component of the robot pose. An early attempt to solve the SLAM problem as a graph-based optimisation problem was reported by Lu and Milios (1997), who introduced a method based on a maximum likelihood criterion to optimally combine all the spatial relations between all the local frames of sensor data. One of the most successful graph-based optimisation methods was introduced by Olson et al. (2006), who presented a method based on stochastic gradient descent, which provides an accurate optimisation for any constraints in the graph. Building on that, and in order to improve performance, Grisetti et al. (2007b) proposed a tree-based parametrisation, which was shown to deliver a significant boost in performance. In addition, the approach can deal with arbitrary graph topologies and can also perform 3D optimisation.

Different vision-based mapping methods using graph optimisation have been proposed (Andreasson et al., 2008; Eade and Drummond, 2007; Fraundorfer et al., 2007; Konolige et al., 2010). These methods follow the same general approach whereby the map is built as a graph, with the nodes containing camera views from

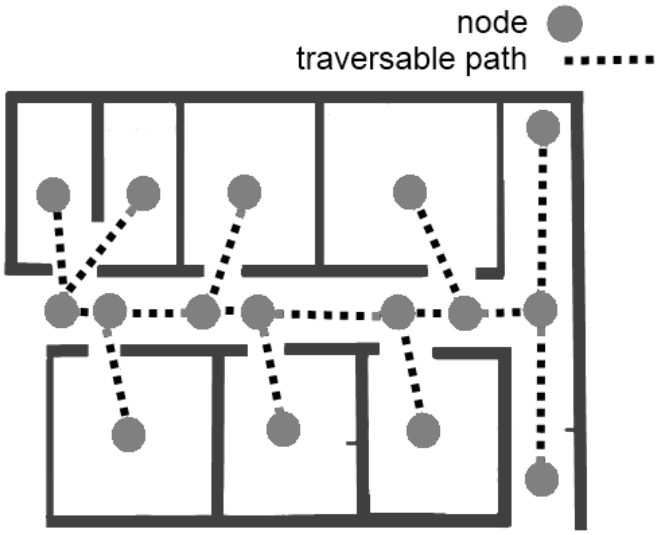

Figure 2.3: An illustration of a topological map for a typical indoor environment.

the environment and the graph edges expressed as geometric constraints between these views. A loop-closing mechanism is deployed to detect when previously mapped areas are revisited. When a loop is detected, a new constraint link is added to the graph and then a graph optimiser is invoked to correct the map. The loop detection mechanism could simply be a direct one-to-one view-matching procedure; however, more sophisticated methods for loop detection can be used to provide real-time performance when the graph contains a large number of nodes. Such methods include the hierarchical vocabulary tree (Nister and Stewenius, 2006) and the FAB-MAP visual vocabulary (Cummins and Newman, 2008).

2.2.2 Topological Methods

Generally, in this type of mapping methods, the world is represented as a graph where nodes correspond to certain places in the environment and the edges between these nodes represent traversable paths between these places, as in Fig. 2.3.

Compared to metric mapping methods, where the elements of the mapped environment are represented to scale, topological mapping provides a more abstract representation of the world, leading to more scalable and less storage-demanding maps. Topological maps are especially well suited for tasks where human-robot interaction has a high priority, as they are an intuitive and natural way of representing the world to humans. This assumption is supported by studies from the field of cognitive mapping, where research indicates that topological representations are used by humans to represent large and complex environments (Lynch, 1960; Siegel et al., 1975).

An important step in the topological mapping process requires a transformation of the robot's sensory measurements (e.g. laser scans, sonar scans, odometry, image features) to an abstract representation of the world including a set of nodes and connections between these nodes. Different mapping implementations have been proposed to perform this abstraction step, but they differ in their definition of the graph and how to select the location of the nodes in the map and the information which is stored in the connection between the nodes. Some examples include (Choset and Nagatani, 2001; Kuipers and Byun, 1991; Mataric, 1992; Morris et al., 2005; Ranganathan and Dellaert, 2011).

Focusing on those methods that use computer vision as the main sensor for the robot, an early approach for topological mapping can be found in the work of (Ulrich and Nourbakhsh, 2000), where the mapping process was performed in two stages: off-line and on-line. In the off-line stage the robot is driven through its operational area to learn a model of the environment, by manually selecting a set of images captured in certain places and then creating a topological map from these images, which represent the nodes of the map. In the on-line stage the robot

uses the map to find its current position, i.e. the node which is most similar to the current view. Many researchers have used various methods to create the map automatically, without the need for manually selecting the nodes. Zivkovic et al. (2007) formalised the topological mapping problem as an approximate solution for a graph cut problem, whereas Valgren et al. (2006) used an incremental clustering approach to build the topological map, where a set of images are clustered based on an image similarity measure. For each of the produced clusters, one key image is chosen to represent a node in the map.

Probabilistic approaches have been applied to deal with the uncertainties of sensory measurement during the mapping process, especially in the case of building a topological map in an environment high in perceptual aliasing. Perceptual aliasing occurs when different parts of the environment appear the same physically when viewed from the camera on board the robot. This property makes it difficult for a robot to detect when it revisits an already mapped place (loop closing). Some of the probabilistic approaches found in the literature include the work by Goedemé et al. (2007), who applies Dempster-Shafer's probabilistic theory to topological map construction in environments with self-similarities. Also, we need to consider the work by Cummins and Newman (2008), who built a probabilistic system, known as FAB-MAP, which calculates the probability that an observed image comes from any previously visited node in the map, as well as the probability that the image is from a new mapped location in the environment.

Although a topological map does not contain any metric information, different approaches have been proposed to use it for tasks such as visual navigation. Most of the methods for visual navigation using topological maps are derived from the snapshot model for visual homing (Cartwright and Collett, 1983) where the goal of

the navigation is represented using a snapshot of the surrounding visual scene. In the case of a topological map, the nodes are used as an intermediate goals and the navigation is performed as a succession of homing steps. For example, Labrosse (2007) proposed an appearance-based navigation method based on 2D image comparisons. Fraundorfer et al. (2007) used visual odometry Nister et al. (2006), based on local geometric information extracted from neighbouring nodes in the map, in order to estimate the robot displacement between these nodes. Booij et al. (2007) built their navigation system based on heading estimation to achieve hill-climbing behaviour, while Goedemé and Van Gool (2009) presented a system based on wide-baseline image feature matching, which enabled the robot to follow a pre-recorded sequence of omnidirectional images.

However, using topological maps for navigating can only provide approximate directional information for the robot about its current location and the target location which it needs to move towards. When the environment is too large to be represented by a single metric map, topological maps can provide an answer, but when precise navigation and localisation are also needed, a hybrid map, which contains both topological and metric maps, becomes necessary.

2.2.3 Hybrid Methods

As metric maps provide a dense representation of the environment and are particularly suitable for precise trajectory planning, they do not scale well to large-scale environments. On the other hand, pure topological maps provide a higher level of representation, which allows symbolic and goal-directed planning tasks to be performed. The resulting maps are also more compact and can scale better with

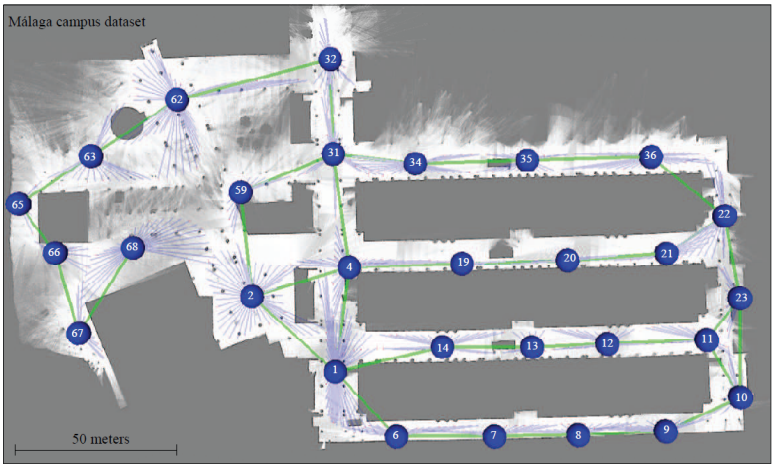

Figure 2.4: Hybrid Metric-Topological Map (Blanco et al., 2008).

the size of the environment, although they cannot be used for accurate navigation. These complementary strengths and weaknesses of metric and topological methods are a motivation for a hybrid mapping approach. In a hybrid map the environment is typically represented by a global topological map, which connects local metric maps (sub-maps).

In general, the main idea behind hybrid mapping can be described as follows. The world is represented on a topological level by a simple graph consisting of connected nodes, whereas on the metric level of the map each node is represented using a relatively small metric map. The main emphasis is on creating metric maps which are located in certain places in the environment and cover the sensor horizon of the robot at these places. Examples for such maps can be found in (Beeson et al., 2010; Bosse et al., 2004; Tomatis et al., 2003). One of the most comprehensive approaches to hybrid mapping was presented by Kuipers (2000), where the environment of the robot was mapped in a hierarchical fashion, referred to as the "Spatial Semantic Hierarchy" (SSH) which consists of several layers of topological and metric maps. Later, (Kuipers et al., 2004) presented an

extension to the spatial semantic hierarchy, using an occupancy grid to represent local metric maps of small-scale space created using a particle filter method, while topological methods have been used to represent the structure of large-scale space. The proposed method creates a set of topological map hypotheses and can handle multiple nested, large-scale loops. Recently, Blanco et al. (2008) formulated the hybrid mapping problem as a unified hybrid metric-topological (HMT) Bayesian estimation via the factorisation of the general SLAM problem (Thrun et al., 2005). This means that the global map produced by the SLAM system is divided into a set of metric sub-maps, which can then be estimated theoretically from conditionally-independent sequences of observations, as shown in Fig. 2.4.

Hybrid mapping using vision as the main sensor has gained increasing popularity in recent years, especially using omnidirectional cameras. Murillo et al. (2007a,b) introduced a hybrid map where, on the topological level, the world is represented using a set of omnidirectional images referred to as the reference views, while on the metric level a group of image features is extracted from each reference view and used to provide metric information about the local position of the robot relative to the visited reference view. This metric information is then extracted using multiple-view geometry. In this thesis, the same hybrid approach for mapping the environment is adopted whereby the world is represented on two levels, global and local. The global level is an adjacency graph of nodes where each node on the local level of the map is represented by a sphere of image features extracted from an omnidirectional image captured in the position of the node. This spherical representation of the nodes creates a connection between the global and local levels of the map. A group of image features is used as a qualitative descriptor for localising the robot to one of the nodes on the global

level of the map, and the 3D location of these features on a sphere is used for estimating the heading needed for the navigation system of the robot on the local level of the map. More details about the structure of the map representation are presented in Chapter 4.

2.3 Robot Mapping in Non-Static Environments

The previously mentioned mapping approaches represent the vast majority of the literature, where the environment is assumed to be static and does not change over time; any changes are normally ignored or considered as measurement outliers. However, in recent years, as the computational power for mobile robots and the availability of inexpensive and reliable sensors has increased, the focus of the mobile robotics community has shifted toward mapping real, everyday environments and performing long-term operations inside human-populated places.

Some properties of the human-populated environments introduce challenges for existing mapping methods. For example, the appearance of a room in a house is not static over time, as new objects are sometimes added, existing objects like pictures or carpets may be changed or moved, and old objects may be removed. Some of these changes occur very often, such as moving chairs and cushions, so they need to be excluded from the long-term representation of the appearance of the environment.

Different approaches have emerged in the literature to deal with these challenges. In the following sections, these approaches are divided into two general categories. The first contains methods which focus on tracking and filtering out rapid environmental changes, such as moving people or cars within an otherwise

static environment. The second category is dedicated to methods which perform continuous mapping where the mapping process is carried out infinitely.

2.3.1 Environments with Underlying Static Structure

A naive way to map a non-static environment could be by simply assuming that the underlying structure of the environment is static, and then try to separate moving objects from stationary parts by treating the dynamic effects as measurement outliers (Dong et al., 2007). However, ignoring sensory information about the moving objects in the environment could affect the overall performance of the mapping system when working in highly dynamic environments. Alternatively, many authors try to improve the robustness of the mapping process by detecting and tracking moving objects separately (Anguelov et al., 2002; Migliore et al., 2009; Ortega, 2007; Wang et al., 2007, 2003).

Considering that the environment consists of static and dynamic objects, other approaches build two maps, one for the static parts and one for dynamic elements. The complete state of the environment is obtained by merging the two maps (Wolf and Sukhatme, 2005). Other approaches try to maintain one map for both dynamic and static landmarks, by classifying landmarks as dynamic or stationary using a probabilistic framework. Movements of the dynamic landmarks are observed and included in the estimation process of the map (Bibby and Reid, 2007).

Other authors have proposed an approach to map non-static environments that involves learning a set of configurations and possible states for the dynamic objects in the environment, e.g. corresponding to open and closed doors (Mitsou

and Tzafestas, 2007; Stachniss and Burgard, 2005). This approach makes a strong assumption about the repeatability of a set of a limited number of distinguishable configurations in the mapped environment. This assumption does not hold in many real environments, though, where the number of configurations which result from human activities is unpredictable and not limited to a finite set.

In general, while these approaches can handle some problems of mapping a non-static environment, they cannot deal with long-term changes to the structure of the environment, as they do not provide a method to remove ("forget") old landmarks which have disappeared. One way to handle these types of changes is to re-map the environment from scratch every time a significant change happens, which would lead to discontinuity in the work of the robot and to a loss of experience that the robot has acquired using the old map.

2.3.2 Continuous Mapping

One approach to solving the problem of mapping a non-static environment is to never assume that the map is complete. In other words, knowing that the state of the environment is non-static, the mapping process is carried out continuously, thus adding new information to the map every time the robot enters a new place or re-visits an already mapped area in the environment.

Milford and Wyeth (2010) presented such an approach for mapping non-static environments. They developed a biologically-inspired mapping system based on rodent hippocampus models, known as RatSLAM. The system consists of a pose cell network, which encodes the robot's location and orientation state. The pose cell network is a three-dimensional attractor neural network in which each neuron

simultaneously excites and inhibits its neighbours when activated from a local view cell. The local view cells contain visual templates about each location in the environment visited by the robot, as well as an activation threshold for activating the cell when the template matches the current view of the robot. The output of the RatSLAM system is a graph-based map where information from the view cells and pose cells, as well as the odometry, is combined to create the nodes and the links between them in the map. The map is called the "experience map". The system continually adds nodes to the experience map as the robot explores more places in the environment and as the geometry and visual appearance of previously mapped places change over time. In order to keep the size of the map manageable, an upper bound constraint is used to trigger a graph pruning process, which limits the size of the experience map.

Konolige and Bowman (2009) adopted a similar approach, where they used a mapping system called FrameSLAM (Konolige and Agrawal, 2008). FrameSLAM builds a graph-based map for the environment consisting of connected image frames. The system continually adds or removes image frames from the map every time the robot re-visits an already mapped area in the environment. Their continuous mapping system keeps adding new frames to the map until a threshold is reached which triggers a deletion process that removes the oldest frames from the map.

Hochdorfer and Schlegel (2009) addressed the problem of growing map size during a life-long mapping operation. Their system builds an image feature-based map in an EKF estimation framework. The number of image features in the map is limited using an upper limit bound. When the number of features reaches the predefined upper limit, the features that are less beneficial for localisation of

the robot are removed from the map. To determine the usefulness of a feature for localising the robot, a k-means clustering of feature positions observed from neighbouring robot poses is performed first, and then in each cluster the features with the lowest information content are removed. The information content of a feature point is calculated using the covariance matrix of the position of the robot from which the feature was observed.

Generally, the above-mentioned approaches handle the problem of mapping non-static environments by continually adding new information to the map until a certain size threshold is reached, which is important for limiting the growth of the map infinitely. After the threshold is reached, a process of removing information from the map is triggered.

The problem with this type of mapping system is that the removal of the information comes only as a response to the growing size of the map. This means that if the map stays below the size threshold, no information will be removed, which in turn will keep old and absolute information stored in the map and could consequently affect the performance of the robot. Therefore, a more efficient solution should be adopted that applies a more sophisticated and selective approach to remove, or "forget", information from the map.

Several authors have investigated mapping systems that incorporate simple forgetting mechanisms based on recency weighting. For example, Yamauchi and Beer (1996) developed a so-called adaptive place network, where the environment of the robot is modelled as an adaptive topological map. Their system assigns a confidence variable for each link in the map that decreases or increases based on the successful and unsuccessful attempts by the robot to traverse the link. Links with low confidence are deleted from the map.

Andrade-Cetto and Sanfeliu (2002) proposed a method to measure the quality and strength of image features in a feature-based map. The strength of an image feature is measured based on how frequent it has been detected by the robot. The quality of an image feature is measured using the covariance of its estimated position in the map through an EKF estimation framework. During localisation tasks, and when the robot revisits an already mapped area in the environment, the strength and quality of the features detected in that area are updated. When the strength and quality of an image feature drops below a given threshold, the feature is removed from the map. Any new detected features are initialized directly into the map. The system was tested inside a small environment with only 50 image features, and no indications about the dynamic nature of the environment were given.

Luo et al. (2007) introduced an adaptive visual place recognition system that was able to recognise the nodes of a topological map based on the appearance of the environment. The system is fully supervised and requires a learning step. During the learning, an incremental support vector machine (SVM) (Cristianini and Shawe-Taylor, 2004) is trained using a collection of images captured inside each node. This learning process is continued over the life time of the robot.

Biber and Duckett (2005) introduced a long-term mapping system for a mobile robot equipped with a laser range scanner and odometry. The environment is represented as a set of nodes and their interrelations. Each node is represented by multiple sets of local occupancy grid maps created from laser scans recorded at the position of the node. These occupancy maps represent the environment at different time scales. When the robot revisits one of the nodes in the map, the occupancy grid map which is most likely to represent the current laser scan

reading from the robot is used as a representation for the node. Adaptation to changes in the environment is achieved by adding new information to the nodes at different rates, which also removes old information from each subset of laser scans in the nodes at different fading rates. However, maintaining multiple occupancy grid maps, one for each different update rate, could become a challenge due to high computational costs when dealing with very large-scale environments.

Walcott (2011) recently proposed a method to update a laser-based metric map. The method uses a pose graph SLAM approach to optimise the trajectory of the robot and produce the map. In order to update the map, the robot compares its current laser scan view with scans stored from previous passes through the same sections of the environment. The robot detects changes in the environment when the difference between the robot's current laser scan and previously stored laser scans exceed a certain threshold. The change is labelled either as "new added objects" or "removed objects". The author makes the assumption that the environment contains only low dynamic objects, i.e. objects that move only outside of the robot's view, which makes the environment static when the robot is passing through it. This assumption is needed for the scan matching process that detects changes in the environment and adds them to the map; however, in a dynamic environment, changes can occur while the robot is operating inside the environment. As such, these changes need to be detected and filtered out.

The long-term adapting mechanism introduced in this thesis is inspired by the multi-store model of human memory, which allows the robot to detect when a change occurs to the appearance of the environment and also provides a mechanism to select only stable landmarks to become part of the map. One of the advantages which makes the approach presented in this work different from the

above-mentioned approaches is that it prevents adding newly detected landmarks to the map directly. The multi-store updating mechanism ensures that spurious landmarks should be quickly forgotten, while only persistent landmarks will be used by the robot for localisation and navigation. More details about the updating mechanism are presented in Chapter 5.

Chapter 3

Basic System Components

This chapter presents necessary background information about the basic components of the system introduced by this thesis. The chapter starts by reviewing different methods for measuring the similarity between images, and motivates the choice of measuring image similarity using local image features. Then it introduces the structure of the omnidirectional vision system used on board the robot and its calibration process. The chapter also presents the method used for estimating epipolar geometry from wide baseline stereo images. Finally, the experimental platform is introduced, along with preliminary experiments.

3.1 Measuring Similarity Between Images

Measuring the similarity between two images plays an essential role for mobile robots using vision-based systems for mapping, localisation and navigation. An important application of this ability for the mapping system is the detection of loops in the map. When the robot re-visits a previously mapped place in the

environment, it is essential to detect that the place has been visited before; otherwise, the robot will assign two distinct parts of the map for the same place in the environment, thus resulting in a corrupted map. The robot can discover such situations by finding a high image similarity between two images taken from the same place (Valgren et al., 2006; Zivkovic et al., 2007). For visual localisation systems, the image captured by the robot's camera is used to localise the robot to the area that is most visually similar in the map (Fraundorfer et al., 2007; Konolige et al., 2010). For visual navigation systems, image feature correspondences between a view from the robot's camera and the most similar view stored in the map can be used to provide the robot with navigational directions toward its goal (Booij et al., 2007).

3.1.1 Local and Global Image Features

Generally, approaches used to measure the similarity between images can be divided into two groups. The first includes methods that measure similarity by comparing global properties between two images. Such global methods include colour histograms (Gross et al., 2003), principal component analysis (PCA) (Vlassis et al., 2002), Fourier transform (Menegatti et al., 2004) and integral invariants (Wolf and Sukhatme, 2005). The second group includes methods that detect and compare local salient features between two images. The term "local" here means that the features are detected from local sub-regions in the image. These methods can be further divided into two main categories based on the type of detected local salient features: corner and region detectors. Some examples of corner detectors include Harris (Chris and Mike, 1988) and the high-speed

corner detector (Rosten and Drummond, 2006). Examples of region detectors include Hessian-Laplace (Mikolajczyk and Schmid, 2001), Scale Invariant Feature Transform (SIFT) detector (Lowe, 1999) and Maximally Stable Extremal Regions (MSER) detector (Matas et al., 2004). Li and Allinson (2008) provides a comprehensive review of these local features for computer vision application.

The first group of methods, i.e. those that use global properties to measure the similarity between two images, are fast, which makes them suitable for applications that require real-time performance. However due to the use of information from the entire image, the global features are vulnerable in situations when there are occlusions and severe viewpoint changes between the two images. This comes from the fact that measuring the similarity between two images using the global methods require first an alignment step (Labrosse, 2006), which can be compromised if there is a significant change between the two images. Whereas, the problem does not appear when using the local image features as there is no need to align the two images before measuring their similarity. The second group, i.e. those that measure similarity using local image features, demand more computational resources and time, but they are shown to be more reliable and robust to occlusion, illumination and viewpoint changes (Deselaers et al., 2008).

The robustness of the local methods comes from the fact that each local image feature is described by a high-dimensional vector, which has high invariance to image translation, scaling and rotation and partial invariance to illumination changes and affine projection. As a consequence of these properties, local methods have gained increased popularity in different computer vision and robotics applications including robot mapping, localisation and navigation (Booij et al., 2007; Se et al., 2002), object recognition (Lowe, 1999) and image retrieval (Nister

and Stewenius, 2006).

In general, methods that extract local image features consist of two steps – a feature detection step followed by a step to build the feature descriptor. Although local methods have many advantages, a common weak point for most of them is that their computation costs are high, in either or both the extraction and description steps. A direct consequence of this fact is the limited use of these features in applications which require real-time performance. In order to overcome this limitation, several approaches have been proposed to accelerate the feature detection and/or description steps (Li and Allinson, 2008).

In this thesis, a local image features detector and descriptor algorithm introduced by Bay et al. (2008), called SURF (Speeded Up Robust Features), is used because it provides fast detection and description performance, which is achieved by introducing approximation to the popular Scale Invariant Feature Transform (SIFT) algorithm (Lowe, 1999), which in turn leads to the features being computed and compared much faster. The following section gives an overview of the SURF feature detector.

3.1.2 Speeded Up Robust Features (SURF)

The SURF feature detector algorithm belongs to the category of region detectors. It can be considered as a speeded up variant of the SIFT detector algorithm (Lowe, 1999), which aims to find local intensity extrema in the image scale space. The scale space representation characterises the input image as a pyramid of different scales, where the image is repeatedly convolved with a second order Gaussian derivative operator and subsequently sub-sampled in order to achieve a higher

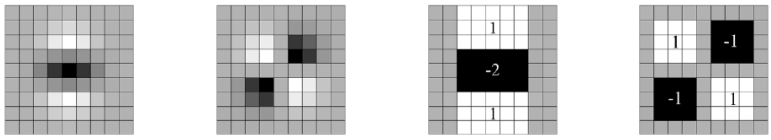

Figure 3.1: Left: Gaussian second order partial derivatives in y-direction and xy-direction. Right: the approximations of the Gaussians as box filters. The grey regions are equal to zero (Bay et al., 2008).

pyramid level. Local extrema in the scale space are invariant to the scale change of the image. In order to speed up the detection process, SURF approximates second order Gaussian derivatives with box filters, see Fig. 3.1, which can be computed rapidly by using integral images (Viola and Jones, 2001).

After the extraction stage, the algorithm creates vector descriptors for the extracted features using information from the local surrounding area. The descriptor is a representation of the Haar-wavelet responses within the feature neighbourhood. Integral image representation is used to speed up the computation of the responses. The result is a descriptor of 64 dimensions, which reduces the time for feature computation and matching when compared with the 128 dimensions of the SIFT feature descriptor vector.

The similarity between two images can be measured by finding the number of correspondences between two groups of the image feature points extracted from each image. These correspondences are found based on the distance measured between the feature descriptor vectors. The number of matched points is then used as a similarity score between the two images; the higher the score, the more similar the two images. However, in situations where the robot is required to compare an image captured by its camera against a database of previously stored images, the matching process can be time-consuming if the number of images is very large

and the search is done linearly. To speed up the search process, some authors propose employing a tree representation of the descriptor vectors extracted from all the images in the database, and then performing a best-bin-first search (Beis and Lowe, 1999). Others propose using a visual vocabulary (Cummins and Newman, 2008; Nister and Stewenius, 2006), which is produced by clustering all the descriptors of the database images. The result is a "bag" of "visual words" which collectively describe the images.

For the work in this thesis, the similarity score between two images is estimated using the number of corresponding image features between two groups of features extracted from each image. The corresponding features are found in the two images based on a nearest neighbour-matching scheme, which is carried out as follows. Each feature point in the first image is compared to each feature point in the second image by calculating the Euclidean distance between their descriptor vectors. A matching pair is detected if its distance is closer than 0.7 times the distance of the second best match, following (Bay et al., 2008). The value 0.7 is a parameter in the nearest neighbour strategy used by the creator of SURF (Bay et al., 2008). This parameter can be tuned based on the desired behaviour for the matching process. The value 0.7 was shown to give good true positive matching results. However, if the target is less false positive, a greater threshold can be used which can also cause the loss of some of the true positive matches.

The similarity score, S_{ij}, between two views, C_i and C_j, can be defined as the number of matched points between them, M_{ij}, or as the percentage:

$$S_{ij} = \frac{M_{ij}}{K_i} * 100, \tag{3.1}$$

where K_i is the number of features in view C_i.

3.2 Omnidirectional Vision System

The vision sensor used in this work is an omnidirectional camera, which was chosen because of the various advantages it provides over a conventional camera (pinhole). The main advantage is the 360^o field of view, which enables the robot to sense the whole surrounding environment in one snapshot, regardless of its heading. In addition to the wide field of view, the omnidirectional camera provides a solution for a well-known problem in computer vision, where motion estimation algorithms mistake a small pure translation in the movement of the camera for a small rotation when the field of view is narrow and/or the depth variation in the scene is small (Dannilidis and Nagel, 1993). This problem is less likely to happen when an omnidirectional camera is used, as the motion estimation algorithms will receive more information for the same movement of the camera than obtained by a conventional camera (Svoboda and Pajdla, 2002).

3.2.1 Structure of the Omnidirectional Camera

An omnidirectional camera can be constructed in different ways, such as rotating a conventional lens system or by using a fish-eye lens to provide the wide-angle field of view. Both of these methods have disadvantages when used in robotic applications because the rotating system requires a considerable time to create the panoramic image, while fish-eye lenses are difficult to design and the commercially available ones are expensive.

Another method used to obtain omnidirectional views is achieved through

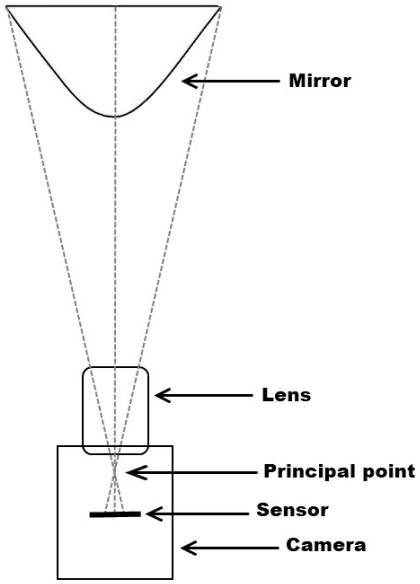

Figure 3.2: Omnidirectional catadioptric imaging systems can be constructed by combining a conventional camera (pinhole) with a convex mirror.

catadioptric imaging systems. As shown in Fig. 3.2, catadioptric imaging systems use a reflecting surface to generate a wide field of view. These type of systems can be constructed by combining a conventional camera with a convex mirror (Nayar and Baker, 1997; Yamazawa et al., 1993; Zivkovic and Booij, 2005). Such systems have many advantages including the 360^o field of view, real-time image capture rate, compactness and low-cost design.

When building omnidirectional catadioptric cameras, four types of mirrors can be used, namely spherical, conical, parabolic and hyperbolic (Ishiguro, 1998). Depending on the shape of the mirror, two types of camera can be designed – the spherical and conical mirrors yield non-central catadioptric cameras, whereas the parabolic and hyperbolic mirrors result in central catadioptric cameras.

Central catadioptric cameras have a single projection centre, on which all

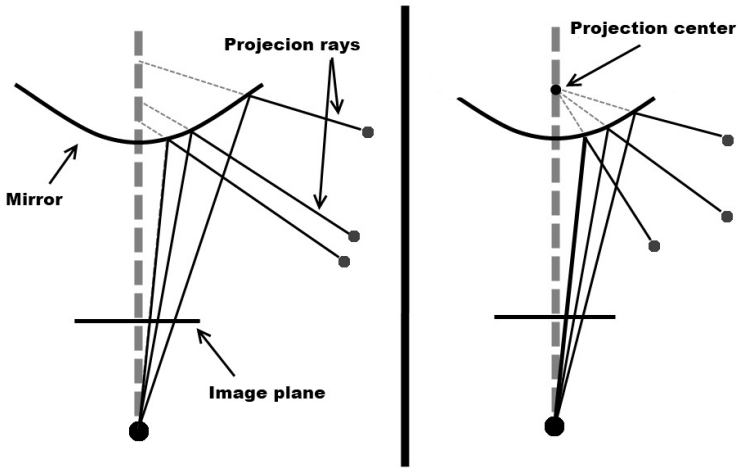

Figure 3.3: Non-central (left) and central (right) omnidirectional systems.

projection rays meet, whereas the non-central catadioptric cameras may have completely unconstrained projection rays (Baker and Nayar, 1999). Fig. 3.3 illustrates the difference between central and non-central cameras.

When using an omnidirectional vision sensor for applications such as mobile robots, it is very useful to choose central catadioptric cameras because the optical centre of projection allows us to apply the same perspective geometry for conventional cameras, since all scene points can be seen from a single viewpoint. Therefore, the classical epipolar geometry for conventional cameras can be generalised to the omnidirectional camera and be used for the same applications (Svoboda and Pajdla, 2002).

For our system, a central catadioptric camera has been assembled by combining a commercially available hyperbolic mirror, acquired from 0-360.com, with a high resolution GigE progressive camera (Jai TMC-4100GE, 4.2 megapixels). The mirror is shown in Fig. 3.4, and has a $115^o \times 360^o$ field of view.

Before the omnidirectional camera can be used, the system needs to be cal-

Figure 3.4: The hyperbolic mirror used in our system.

ibrated in order to estimate its internal parameters. Different methods of calibration have been presented in the literature for both perspective (Zhang, 2002) and catadioptric cameras (Frank et al., 2007; Kang, 2002; Ying and Hu, 2004). The most common approaches are based on the correspondence of image feature points, along with their 3D positions. These methods require as input a set of images taken at several angles and distances to a known calibration object. To calibrate the system, a method developed by Scaramuzza et al. (2006) is used which, as the author showed, gives very good calibration results even if the camera optical centre is not exactly in the focus of the mirror.

3.3 Camera Model and Epipolar Geometry

In this section, relevant aspects of epipolar geometry, which is used for the work in this thesis, are presented. The imaging model that is used for the camera is presented first, and then the epipolar geometry associated with two catadioptric views based on the imaging model is outlined.

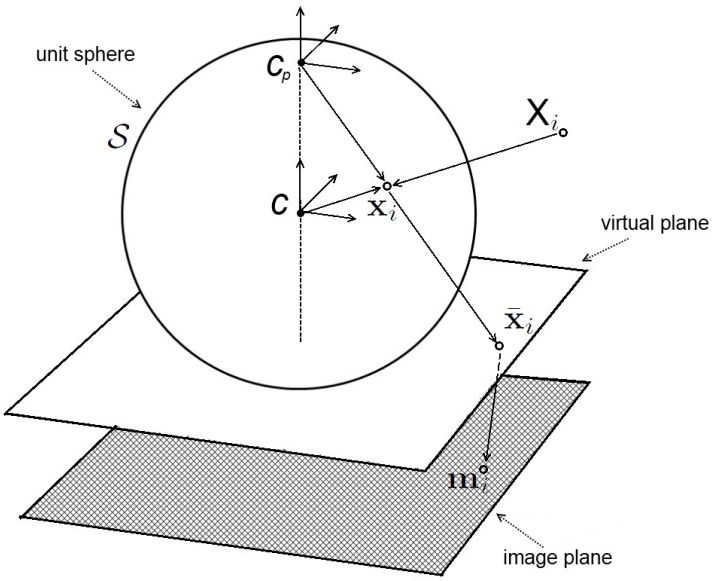

Figure 3.5: Central catadioptric camera models are equivalent to projective mapping from a unit sphere whose centre coincides with the focal point of the curved mirror (Geyer and Daniilidis, 2000).

3.3.1 The Unifying Projection Model

In general, the projection model for any catadioptric system depends on the shape of its curved mirror, which is defined by a set of geometrical parameters. However, these parameters are not always known, such as in our case where the manufacturer of the used mirror does not provide parameter values. In this case, a general projection model can be used if the catadioptric system is central.

Geyer and Daniilidis (2000) introduced a unifying theory for central catadioptric systems and showed that all their projective models are equivalent to projective mapping from a unit sphere, whose centre coincides with the focal point of the curved mirror, to a virtual horizontal plane. As shown in Fig. 3.5, a scene point X_i is projected onto a point $\mathbf{x}_i$ on sphere S. For every $\mathbf{x}_i$ there is

a ray $\bar{\mathbf{x}}_i$ which connects the polar point C_p to $\mathbf{x}_i$ and intersects with a virtual plane above the image plane. Finally, the point on the virtual plane is mapped to a point $\mathbf{m}_i$ on the image plane using the intrinsic calibration matrix of the conventional camera, which is placed under the mirror. Akihiko and Atsushi (2005) adopted the same approach and referred to the unifying projection model as a "spherical camera model", which consists of a camera centre and the surface of a unit sphere whose centre is the projection centre. The surface of the unit sphere is usually referred to as a "spherical image".

3.3.2 Epipolar Geometry for Spherical Cameras

Consider two distinct omnidirectional views of the same 3D scene acquired by a spherical camera (see Fig. 3.6). Let C and C' be the centre of each view. A scene point P can be projected through the two spheres to the centre of projection for each camera. If we let X represent the position of P in the reference frame of C and $\mathbf{x}$ represent the projection of X on the surface of a unit sphere, then it is possible to write:

$$\lambda \mathbf{x} = X, \quad \lambda \in \mathbb{R}_+. \tag{3.2}$$

In the same way for the second camera, the following can be written:

$$\lambda' \mathbf{x}' = X', \quad \lambda' \in \mathbb{R}_+, \tag{3.3}$$

where $\mathbf{x}'$, X' are the 3D points in the frame of C'.

Assuming that $C = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^T$, $\mathbf{R} \in SO(3)$ is the rotation matrix and $\mathbf{t} \in \mathbb{R}^3$

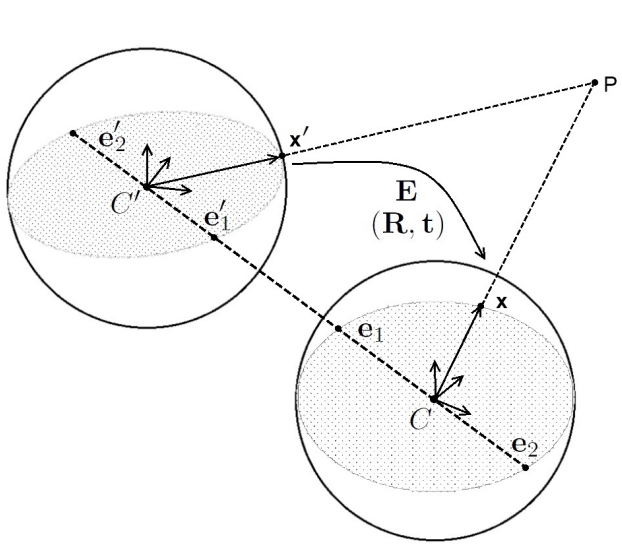

Figure 3.6: Epipolar geometry for spherical cameras using the unifying model of projection (Geyer and Daniilidis, 2000).

the translation direction between C and C', it is possible to write:

$$\lambda' \mathbf{x}' = X' = \mathbf{R}X + \mathbf{t}, \tag{3.4}$$

$$\lambda' \mathbf{x}' = \lambda \mathbf{R}\mathbf{x} + \mathbf{t}. \tag{3.5}$$

According to the epipolar geometry constraints on two views (Hartley and Zisserman, 2004), the following relation can be established:

$$(\mathbf{x}')^T \mathbf{E}\mathbf{x} = 0, \tag{3.6}$$

where $\mathbf{E} \in \mathbb{R}^{3\times3}$ is the essential matrix. Points $\mathbf{e}_1$, $\mathbf{e}_2$, $\mathbf{e}_1^{'}$ and $\mathbf{e}_2^{'}$ are called "epipoles", which result from the intersection of the segment $\overline{C'C}$, referred to as the baseline, with the two spheres. The essential matrix can be linearly solved using eight pairs of corresponding points from the two spheres (Longuet-Higgins,

1981). This matrix can then be decomposed into the rotation matrix $\mathbf{R}$ and the translation skew-symmetric matrix $[\mathbf{t}]_\times$ between the two views as follows:

$$\mathbf{E} = [\mathbf{t}]_\times \mathbf{R}, \tag{3.7}$$

where

$$\mathbf{R} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}, \; [\mathbf{t}]_\times = \begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{bmatrix}.$$

3.3.2.1 Essential Matrix Estimation

In order to estimate the parameters of the essential matrix, the eight-point corresponding algorithm (Longuet-Higgins, 1981) is used. The corresponding points are generated from two views using the SURF descriptors, which will typically generate more than 8 correspondences between the two views. This means that the solution is over-determined, and also due to the fact that false matches will always be part of the matching process, it is not possible to find a value of $\mathbf{E}$ which satisfies the constraints exactly for all points. These two issues can be solved by using the RANSAC algorithm (Fischler and Bolles, 1981), to deal with rejecting false matches, and a total least squares minimizer (Huffel and Vandewalle, 1991), to estimate the value of the essential matrix parameters.

Let $\mathbf{x}_i$, $\mathbf{x}'_i$ be the corresponding points between the two spheres:

$$\mathbf{x}_i = \begin{bmatrix} u_i \\ v_i \\ w_i \end{bmatrix}, \; \mathbf{x}'_i = \begin{bmatrix} u'_i \\ v'_i \\ w'_i \end{bmatrix}$$

and let the essential matrix $\mathbf{E}$ be

$$\mathbf{E} = \begin{bmatrix} e_{11} & e_{12} & e_{13} \\ e_{21} & e_{22} & e_{23} \\ e_{31} & e_{32} & e_{33} \end{bmatrix}.$$

the terms in Eq. 3.6 can be rearranged as follows:

$$\mathbf{Ab} = 0, \tag{3.8}$$

where the rows of $\mathbf{A}$ are

$$\mathbf{a}_i = \begin{bmatrix} u_i u'_i & u_i v'_i & u_i w'_i & v_i u'_i & v_i v'_i & v_i w'_i & w_i u'_i & w_i v'_i & w_i w'_i \end{bmatrix},$$

and the elements of $\mathbf{E}$ are represented by the vector

$$\mathbf{b} = \begin{bmatrix} e_{11} & e_{12} & e_{13} & e_{21} & e_{22} & e_{23} & e_{31} & e_{32} & e_{33} \end{bmatrix}.$$

Eq. 3.8 is a set of homogeneous linear equations, which means that the total least square solution to $\mathbf{b}$ is the right singular vector corresponding to the smallest singular value of $\mathbf{A}$ (Huffel and Vandewalle, 1991).

The RANSAC algorithm provides an efficient way of reducing noise due to outliers and finding the best approximation of $\mathbf{E}$ that fits the data. The essential matrix in our case needs a minimum of 8 corresponding points to be estimated, so RANSAC randomly selects these points from the group of correspondence points. Then, for each selected group, the algorithm estimates the essential matrix using the eight pairs algorithm (Longuet-Higgins, 1981), along with a group of inliers

that are selected from the total group of points based on a geometrical error threshold. In the second stage, the estimated matrix with the largest number of inliers is chosen and all its inliers are used for the estimation of the essential matrix.

In order to evaluate each estimated essential matrix during the intermediate stages of RANSAC, and to find the group of inlier points for each matrix, the Sampson error (Hartley and Zisserman, 2004) is used as a performance metric. Given an essential matrix $\mathbf{E}$, the Sampson error for the i^{th} correspondence($\mathbf{x}_i$,$\mathbf{x}'_i$) is defined as

$$err_i = \frac{(\mathbf{x}_i'^T \mathbf{E} \mathbf{x}_i)^2}{\sum_j (\mathbf{E}\mathbf{x}_i)_j^2 + (\mathbf{E}^T \mathbf{x}'_i)_j^2} \ , \tag{3.9}$$

where $(\mathbf{E}\mathbf{x}_i)_j{}^2$ represents the square of the j^{th} entry of the product $\mathbf{E}\mathbf{x}_i$.

In this work it is assumed that the robot and the omnidirectional camera are working on a planar floor, which means that an additional constraint can be added, which states that the rotation between the spherical views will only be around the vertical axis. The process of estimating the essential matrix can be then simplified by restricting the matrix to the following sparse form (Košecká et al., 2005), assuming translation in the x-y plane and rotation around the z-axis:

$$\mathbf{E} = \begin{bmatrix} 0 & 0 & e_{13} \\ 0 & 0 & e_{23} \\ e_{31} & e_{32} & 0 \end{bmatrix} . \tag{3.10}$$

After a robust estimation of the essential matrix, the rotation matrix $\mathbf{R}$ and translation direction vector $\mathbf{t}$ can be found, as described in the following section. The section starts by presenting the estimation method for the general case and

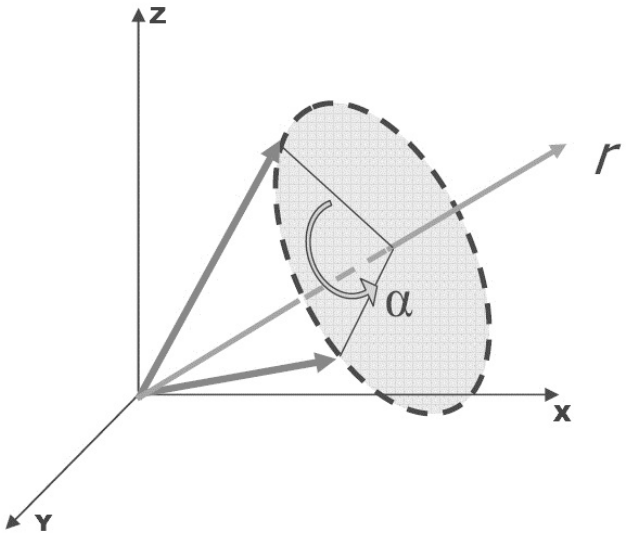

Figure 3.7: Rotation by angle α around unit vector r

then it present the simplified estimation method for the special case where a planner movement assumption is used for the robot. In this work, the system has not been tested against breaking the planner movement where it is assumed that the robot and the omnidirectional camera are working on a planner floor all the time.

3.3.2.2 Estimation of the Rotation and Translation Direction

Based on the method introduced in (Hartley and Zisserman, 2004), the essential matrix can be factored (see Eq. 3.7) into the rotation matrix $\mathbf{R}$ and the skew-symmetric matrix of the translation vector $[\mathbf{t}]_\times$. This method is based on the singular value decomposition of $\mathbf{E}$. The decomposition can be written as follows:

$$\mathbf{E} = \mathbf{U}\mathbf{D}\mathbf{V}^T, \tag{3.11}$$

where $\mathbf{U}$ and $\mathbf{V}$ are orthogonal matrices and $\mathbf{D}$ is a diagonal matrix.

Then $[\mathbf{t}]_\times$ and $\mathbf{R}$ can be computed using the following formula:

$$[\mathbf{t}]_\times = \pm\mathbf{U}\mathbf{Z}\mathbf{U}^T,$$
$$\mathbf{R} = \mathbf{U}\mathbf{W}\mathbf{V}^T \text{ or } \mathbf{R} = \mathbf{U}\mathbf{W}^T\mathbf{V}^T,$$

where

$$\mathbf{W} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \ \mathbf{Z} = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}.$$

This will generate multiple solutions, i.e. four possible combinations of $\mathbf{t}$ and $\mathbf{R}$. However, by applying the positive depth constraint, the one solution where the reconstructed point lies outside of the two spheres (Horn, 1990) can be obtained.

After estimating the rotation matrix, the elements of the rotation movement of the robot (i.e. the angle of rotation and the axis of rotation, see Fig. 3.7) can be extracted using Rodrigues' rotation formula (Faugeras, 1993):

$$\mathbf{R} = \cos(\alpha)\mathbf{I} + (1 - \cos(\alpha))\mathbf{r}\mathbf{r}^T + \sin(\alpha)[\mathbf{r}]_\times, \tag{3.12}$$

where α is the counter-clockwise angle of rotation around the axis of rotation $\mathbf{r}$, $\mathbf{I}$ is the 3×3 identity matrix and $[\mathbf{r}]_\times$ is the skew-symmetric matrix of the axis $\mathbf{r}$.

In the special case when the planar movement assumption is used, the estimation of the rotation between the two views becomes mush simpler, with the

z-axis as the axis of rotation and the rotation matrix taking the following form:

$$\mathbf{R} = \begin{bmatrix} \cos(\alpha) & \sin(\alpha) & 0 \\ -\sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{3.13}$$

3.3.2.3 3D Reconstruction by Stereo Triangulation

After estimating the motion parameters of the camera between two views using the essential matrix, the 3D position of the corresponding points between these two views can be recovered as follows.

Let X' and X be coordinates of a point P in the first and second camera view frames (see Fig. 3.6). Given the rigid motion parameters (rotation matrix $\mathbf{R}$ and translation direction $\mathbf{t}$) between the two camera centres (C', C), the coordinates of P in the two frames are related to each other through a rigid body transformation equation:

$$X' = \mathbf{R}X + \mathbf{t}. \tag{3.14}$$

Let $\mathbf{x}'$ and $\mathbf{x}$ be coordinates of the intersection points between rays $\overrightarrow{CX}$ and $\overrightarrow{C'X'}$ and the unit sphere around each centre of the two cameras. Coordinate vectors $\mathbf{x}'$, X' and $\mathbf{x}$, X are related as follows:

$$\begin{aligned} \lambda'\mathbf{x}' &= X', \\ \lambda\mathbf{x} &= X, \end{aligned} \tag{3.15}$$

where λ' and λ represent the distance between the centre of each camera and the scene point up to an unknown scale factor.

Using Eq. 3.15 in Eq. 3.14, it is possible to write:

$$\lambda' \mathbf{x}' = \lambda \mathbf{R}\mathbf{x} + \mathbf{t}, \tag{3.16}$$

leading to the following relation:

$$\begin{bmatrix} -\mathbf{R}\mathbf{x} & \mathbf{x}' \end{bmatrix} \begin{bmatrix} \lambda \\ \lambda' \end{bmatrix} = \mathbf{t}. \tag{3.17}$$

This formula can also be expressed as:

$$\mathbf{A}\mathbf{b} = \mathbf{t}, \tag{3.18}$$

where $\mathbf{A} = \begin{bmatrix} \alpha & \mathbf{x}' \end{bmatrix}$, $\alpha = -\mathbf{R}\mathbf{x}$ and $\mathbf{b} = \begin{bmatrix} \lambda \\ \lambda' \end{bmatrix}$.
The least square solution for the linear system in Eq. 3.18 is then:

$$\begin{aligned} \begin{bmatrix} \lambda \\ \lambda' \end{bmatrix} &= (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{t} \\ &= \frac{1}{\|\alpha\|^2\|\mathbf{x}'\|^2 - \langle \alpha, \mathbf{x}' \rangle^2} \begin{bmatrix} \|\mathbf{x}'\|^2 & -\langle \alpha, \mathbf{x}' \rangle \\ -\langle \alpha, \mathbf{x}' \rangle & \|\alpha\|^2 \end{bmatrix} \begin{bmatrix} \alpha \\ \mathbf{x}' \end{bmatrix} \begin{bmatrix} \mathbf{t} \end{bmatrix}, \end{aligned} \tag{3.19}$$

where $\langle ., . \rangle$ is the standard scalar product operator.

From Eq. 3.19, λ can be found as:

$$\lambda = \frac{\|\mathbf{x}'\|^2 \langle \alpha, \mathbf{t} \rangle - \langle \alpha, \mathbf{x}' \rangle \langle \mathbf{x}', \mathbf{t} \rangle}{\|\alpha\|^2\|\mathbf{x}'\|^2 - \langle \alpha, \mathbf{x}' \rangle^2}. \tag{3.20}$$

Using this solution, the 3D point $\lambda\mathbf{x} = X$, which is the intersection point of rays $\overrightarrow{CX}$ and $\overrightarrow{C'X'}$ in the reference frame of C, is obtained. However, this is an ideal case, with no noise involved in the process – in a realistic situation there will always be some noise in the system and the two rays will not intersect. In such a case, the above solution will give the point on the rays $\overrightarrow{CX}$, which is the closest one to the ray $\overrightarrow{C'X'}$.

Finally, a note on the scale of the reconstructed points. Rigid motion parameters, which are estimated from the essential matrix, consist of a rotation matrix $\mathbf{R}$ and a translation direction $\mathbf{t}$, which means that only the direction of the translation is known, and not the length of the translation between the two views. The length of the translation is important to produce a 3D reconstruction in the correct scale. When this information is not available, the norm of the translation vector can be set to unity and the 3D position of the reconstructed points then will be determined up to an unknown scale.

3.4 The Experimental Platform and its Verification

Our experimental platform is an ActivMedia P3-AT robot equipped with a high resolution TMC-4100GE camera from Jai, with a curved mirror from 0-360.com (see Fig. 3.8). The camera captures images with a resolution of 2048×2048 pixels. The frame which holds the camera on top of the robot is placed in such a way that the vertical optical axis of the camera coincides with the rotation axis of the robot.

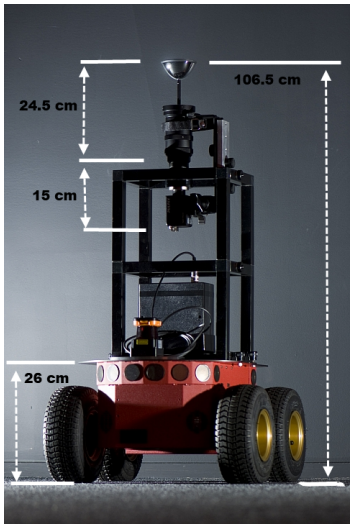

Figure 3.8: The experimental platform: an ActivMedia P3-AT robot equipped with an omnidirectional vision system.

3.4.1 Evaluation of the Essential Matrix Estimation

The following experiment was designed to evaluate the quality of estimating the essential matrix based on the method explained in Section 3.3.2.1. In order to do so, a known relative rotation between two views is compared to that extracted from the essential matrix. The error between the ground truth rotation and the estimated one is used as an indicator for the quality of the estimated essential matrix.

As mentioned earlier, for the work in this thesis, the planar floor assumption is adopted where it is assumed that the robot is working on a horizontal and flat surface. The experiment starts by placing the robot in the centre of a 1.5×1.5 metre square, after which an omnidirectional image is captured and a group of SURF features extracted from the image. This image will be referred to as the "reference image". Next, a spherical image is created by back-projecting the

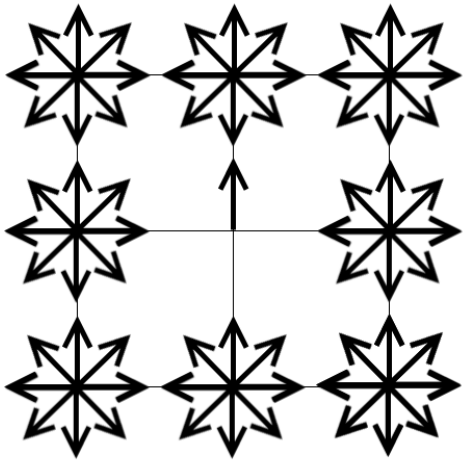

Figure 3.9: A total of 64 different images taken at eight places around a reference image. At each place the robot captures eight images at known relative orientation to the reference image. The images were captured at $0, -45, -90, -135, -180, +135, +90, +45$ degrees. The arrows represent the heading of the robot.

feature points from the image plane to the surface of a unit sphere. After that a group of images is captured from eight locations on the square (see Fig. 3.9). At each location the robot is placed at eight different relative orientations to the reference image, i.e. $0, -45, -90, -135, -180, +135, +90, +45$ degrees. The 8 different orientations act as a ground truth for our experiment. The essential matrix is then estimated between each image on the border of the square and the reference image in the middle, and then the rotation matrix is extracted. The results show that the mean error in estimating the relative orientation is $2.12^\circ \pm 2.10^\circ$ compared to the ground truth which have been obtained manually. This level of error is within the range of the error of estimating the ground truth and considered manageable when the robot uses the estimated rotation to provide directional clues for visual autonomous navigation.

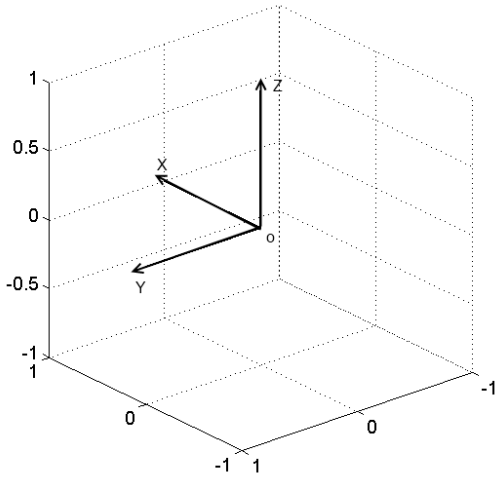

Figure 3.10: The movement of the robot is done on the x-axis, where the y-axis is looking to the left of the robot and the z-axis is pointing upwards.

3.4.2 3D Reconstruction

This experiment tests the 3D reconstruction of a group of correspondences between two images captured inside a room where the 3D positions of a group of manually selected points from the scene are reconstructed. The height of the selected points above the ground is known, along with the relative distances between them. This information is used as a ground truth for evaluation.

Two omnidirectional images are captured from the camera on board the robot. The distance between the two images is set to 1 m and the relative orientation is $0\,^\circ$ degrees, which corresponds to the robot being moved 1 m forward between the two images. The movement is done on the x-axis, where the y-axis is looking to the left of the robot and the z-axis is pointing upwards (see Fig. 3.10).

In the next step, SURF features are extracted from the two images generating two groups of feature points (920 and 858 points, respectively, in our case). Using

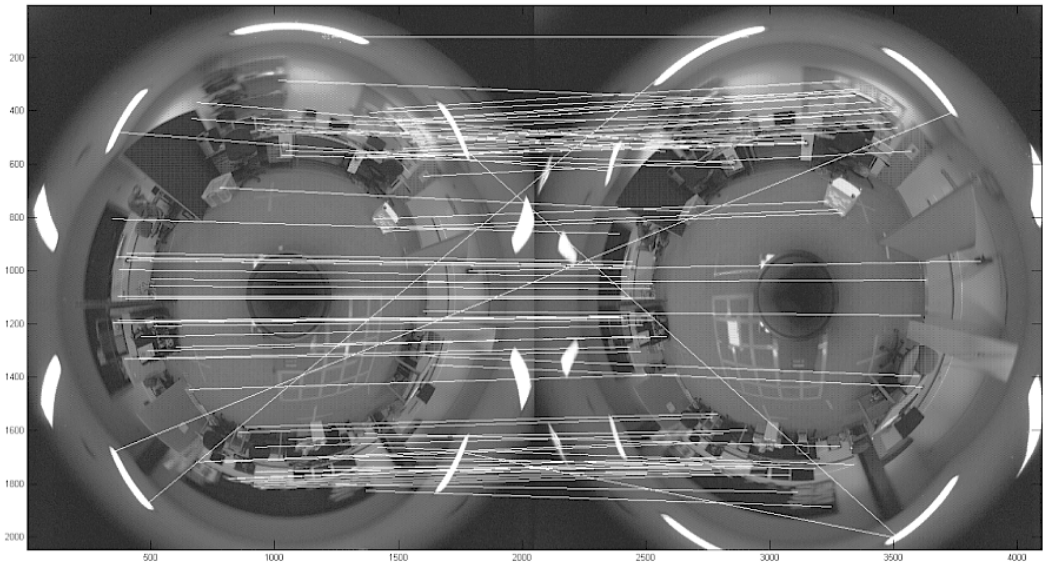

Figure 3.11: A group of matched SURF points between two images. Some of the mismatches are visible, especially from the ceiling where the feature points look similar due to the regularity of the ceiling structure.

the features' descriptors, a group of correspondences is obtained between the two images (203 correspondences in this experiment). Fig. 3.11 shows the matched points between the two images and demonstrates that the matching process is not perfect and mismatches are always possible. To overcome this problem, the RANSAC algorithm is used to estimate the essential matrix, as explained in Section 3.3.2.1. The result is the following matrix with 192 correspondences points as inliers:

$$\mathbf{E} = \begin{bmatrix} 0 & 0 & 0.0084 \\ 0 & 0 & -0.7051 \\ -0.0046 & 0.7091 & 0 \end{bmatrix}.$$

This matrix is then factorised into the following rotation matrix and the skew-

symmetric matrix of the translation vector (see Section 3.3.2.2):

$$\mathbf{R} = \begin{bmatrix} 1 & -0.0054 & 0 \\ 0.0054 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, [\mathbf{t}]_\times = \begin{bmatrix} 0 & 0 & -0.0119 \\ 0 & 0 & 0.9999 \\ 0.0119 & -0.9999 & 0 \end{bmatrix}.$$

Based on the estimated rotation matrix, the relative rotation between the two views is 0.29 ° and the translation direction is $\mathbf{t} = \begin{bmatrix} -0.9999 & -0.0119 & 0 \end{bmatrix}^T$. This result reflects the actual movement of the robot along the x axis.

After estimating the rotation matrix and the translation direction, the 3D positions of the feature points can be estimated based on the method outlined in Section 3.3.2.3. Fig. 3.12 shows the reconstructed points. The room where this experiment took place was 8.30 × 6.90 m in size, and due to the length of the baseline between the two views being 1 m, the reconstruction has the scale of the actual view. By looking at the view from the yz-plane in the bottom-left corner (Fig. 3.12), it is clear that the points are indeed expanding over 6.90 m; however, when looking at the views from the xy- and xz-plane, it is clear that some of the points are reconstructed beyond the border of the room, exceeding 8.30 m.

In general, the reason for this error comes from two sources. The first possibility is that these points coincide with the direction of the robot's movement, thus resulting in poor parallax and an overestimation of the depth (see the top row of Fig. 3.12). The second possibility is that RANSAC was not successful in eliminating all the outliers, which consequently gave an imprecise estimation of the depth.

In order to deal with these cases, the fact that the robot is working inside an

indoor environment is used. In this environment the walls act as a natural barrier for the depth of the reconstructed points. This means that the image feature points will be generated from the objects which exist between the robot and the walls. Therefore, the depth of the reconstructed points will range between the robot and the walls, and any point which has an overestimated depth will appear as an outlier for this distribution, which can be detected by building a distribution histogram for the depth of all the reconstructed points in the scene. The outliers will represent the tail of the distribution. Fig. 3.13 shows the histogram for our case. The points at two standard deviations (or more) away from the mean can be removed, assuming that the distribution of their depth follows a normal distribution and 95% of the points will fall within two standard deviations.

In order to examine the accuracy of the reconstruction process in more detail, 20 points on the floor of the room and six points on the corners of a box located on the floor were marked by hand (see Fig. 3.14). After reconstructing the 3D positions of the points from the floor, the mean height of these points was 1.073 ± 0.009 m. Knowing that the height of the camera mirror above the floor is 1.065 m, the error of the re-construction is 0.008 m on the z-axis. For the box, the length of the edges was 0.30 m, and when six points from the corner of the edges were reconstructed, as shown in the right-hand side of Fig. 3.14, the mean distance of the edges was 0.310 ± 0.007 m. This is an error of 0.010 m compared with the actual measurements.

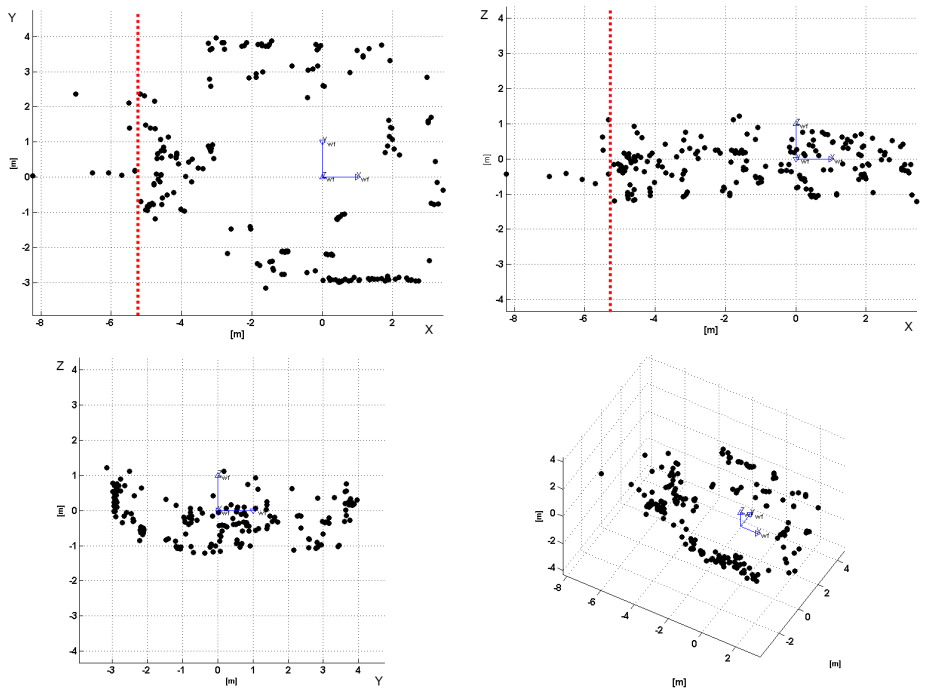

Figure 3.12: Four angle views of the reconstructed points extracted from two images taken inside a room 8.30 × 6.90 m in size. The robot moves by 1 metre parallel to the longer wall of the room. The view from the xy-plane is top-left, the view from the xz-plane is top-right and the view from the yz-plane is bottom-left. The dashed red line represents two standard deviations away from the mean of the distribution of the points' depth.

3.5 Summary

This chapter presented the essential components of the vision system used in this thesis. The chapter described the method by which the similarity between two images was measured using local image features. After that, the structure of the omnidirectional camera used on board the robot was presented along with the multiple view geometry for estimating the essential matrix and the 3D reconstruction of image features. That was followed by preliminary experiments showing the performance of the multiple view geometry estimation. The preliminary ex-

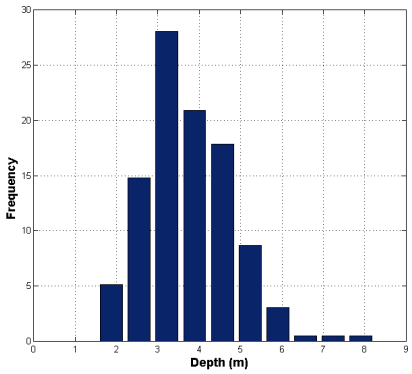

Figure 3.13: Distribution of the estimated depth of feature points between two views. The mean is $3.72m$ and the standard deviation is $1.09m$.

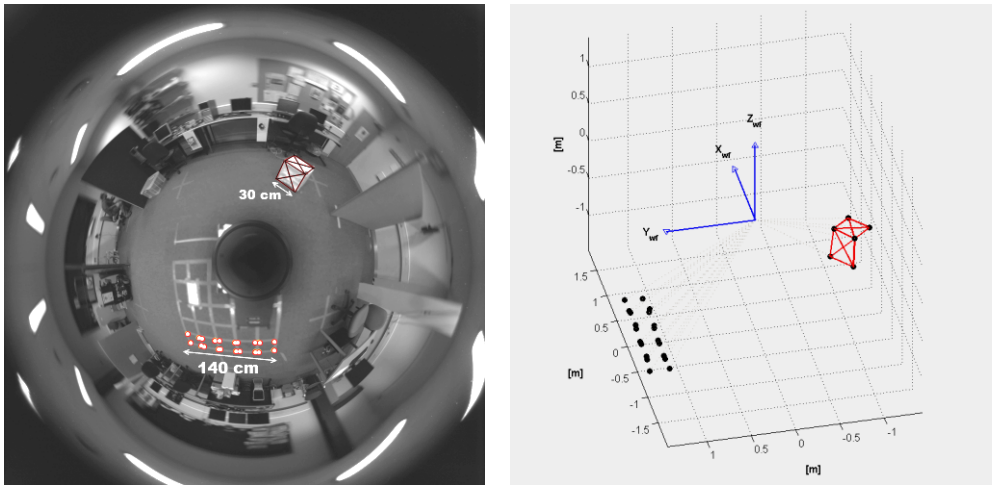

Figure 3.14: Reconstruction of 20 correspondences on the floor, along with six points from the corner of a box. The relative distances between the points are known, as well as the height of the robot's camera above the floor.

periments are aimed to provide the reader with an anecdotal understanding about the multiple view geometry estimation used for the rest of the work in this thesis. Chapter 6 contains the extensive experimental analyses which were conducted to evaluate the proposed long-term mapping system.

Chapter 4

Visual Hybrid Map

This chapter describes the method used by the robot to build a sparse hybrid map of its environment, which consists of two levels – a global level containing a sparse topological map and a local level which stores a spherical view for each node. A graph-pruning technique based on a dual clustering approach is developed to select the nodes on the global level of the map. Multiple view geometry is used to enable the robot to use the spherical views of the nodes as directional way-points while travelling from one place to another.

4.1 Introduction

As described earlier in Chapter 2, different methods have been introduced to tackle the problem of acquiring a map of a mobile robot's environment. The different types of maps are divided into three groups, namely metric, topological and hybrid. This chapter presents a method to enable a mobile robot to build a map that belongs to the hybrid group.

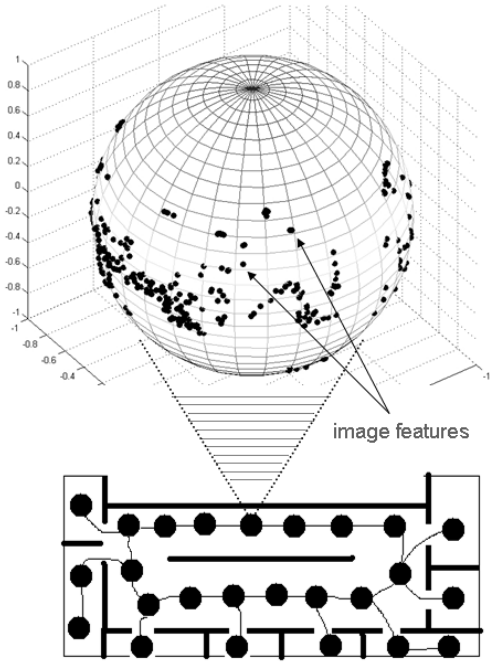

Figure 4.1: The proposed hybrid map with two levels, i.e. global and local. The environment is represented as an adjacency graph of nodes on the global level of the map, and each node on the local level represents the 3D location of image features on a sphere. This method represents the direction of the features (but not their distance or depth) from the centre of the sphere, which corresponds to the centre of that node.

The mapping process is achieved through two steps. First, the robot builds an initial dense pose-graph map of its environment using odometry and vision information. Then, in the second step, the initial map is used to build a sparse hybrid map consisting of two levels, global and local. Fig. 4.1 illustrates the hybrid map. On the global level, the world is represented as a topological map, which is extracted from the initial dense pose-graph using a dual clustering approach, introduced later in this chapter. The proposed approach selects nodes from the initial map which are located in areas such as doorways and corners, allowing the topological map to maintain full coverage of the environment while minimising

the required number of nodes. *The distances between the nodes in the initial dense pose-graph are used only in the process of selecting the nodes of the final global map. When using the map for localization and navigation, the robot does not use any metric information about the position of the nodes.* On the local level of the map, each node stores a spherical view representation of image features. The spherical views contain the 3D location of the image features on a sphere, so only the directions of the features (but not their distance or depth) from the centre of the sphere are stored. The centre of the sphere in this case corresponds to the centre of that node.

Each spherical view is initialised from an omnidirectional image recorded from the centre of each node in the global map. The spherical representation of the nodes creates a connection between the global and local levels of the map, where the group of image features is used as a qualitative descriptor for localisation on the global level, and the 3D location of these features on the sphere is used for estimating the heading needed for the navigation system at the local level.

Localisation on the global level is achieved by using an image similarity score based on the number of matched feature points between the current view of the robot and the group of points stored in each node. The robot localises itself to the node which has the highest similarity score in the map. Navigation on the local level is done by using multiple view geometry for spherical views to estimate the robot's heading during autonomous navigation. This navigation method is described in Section 4.4. Later, Chapter 5 describes how the robot uses these same image features to update the spherical views stored inside each node over time, in response to changes in the appearance of the environment.

A similar spatial representation to the spherical view representation presented

in this thesis can be found in (Peters et al., 2001), where an egocentric view of objects around a mobile robot is represented using a so-called sensory ego-sphere (SES). The SES is a sphere centred at the origin of the robot's coordinate frame and is considered as a database which stores localised sensory information about objects from different locations in the robot's environment. The surface of the SES is constructed as a graph, which divides the space into hexagonal or pentagonal regions. Each node in the graph corresponds to an entry in the database of objects. The SES serves as short-term memory for the robot, wherein data is continuously removed from the sphere after a predefined time limit. The main aim of the SES is to support the robot in cognitive tasks such as attentional processes and spatio-temporal event detection, while the spherical representation proposed in this thesis is used for both short- and long-term representation of a changing environment.

4.2 Building the Initial Map

The initial map-building process is carried out based on a stop-sense-go strategy, where the robot is driven by a human operator. The driver follows the following routine. The robot starts a mapping step by capturing an omnidirectional image from the on-board camera, while it is static, along with the odometry reading from the wheel encoder. Then the robot moves a short distance forward and stops. If there is a need to perform a rotation, the robot rotates a certain angle and ends the current mapping step by stopping. The driver repeats as many mapping steps as required to cover the environment.

The constraints between each consecutive pair of poses along the trajectory

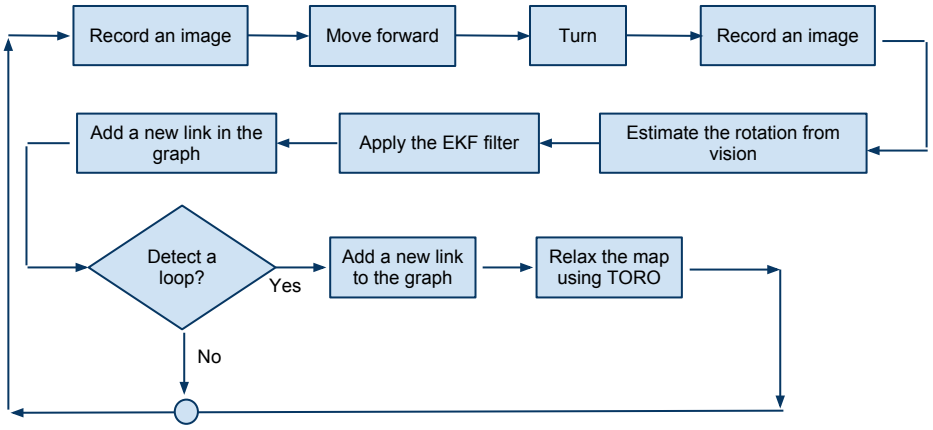

Figure 4.2: The robot starts building a graph-based map by capturing an omnidirectional image, and then performs a translation movement followed by a rotation. The robot then stops and captures another omnidirectional image. After that the robot estimates the executed rotation using the two images captured before and after the rotation. An EKF filter is deployed to correct the rotation measured from the wheel encoder using the estimated rotation from vision. The current and previous poses of the robot are then linked in the graph, after which the robot checks whether or not a loop has been closed from each current position by measuring the image similarity between its current view and the previously stored views in the map within a pre-specified radius. If a loop is detected, a link with a zero distance is added to the graph, and then the graph optimiser algorithm is invoked to correct the structure of the map based on the newly added link.

of the robot consist of two components, namely translation and rotation, which can be extracted from the internal odometry of the robot. However, using the odometry alone is not always accurate enough to build the constraint network. First, odometry measurements are affected by noise caused by wheel drift and slippage, especially during rotation. Second, using odometry alone cannot provide any information about loop closure. To deal with these limitations, measurements from the omnidirectional vision sensor on board the robot are used as an input for a Bayesian filtering framework (extended Kalman filter), in order to reduce errors from the odometry measurements. In addition, the vision sensor is used

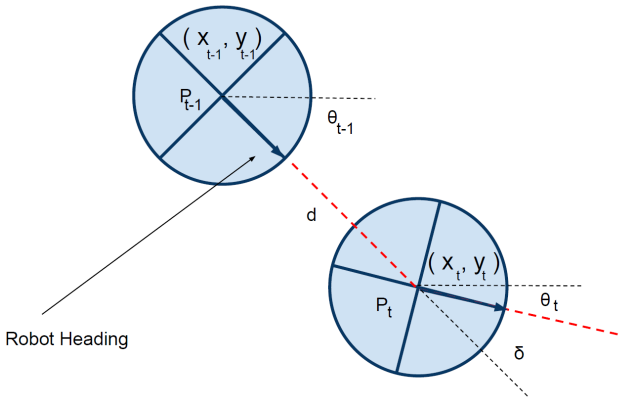

Figure 4.3: Odometry model: The robot motion between two consecutive poses is approximated by translation d followed by rotation δ.

to perform loop closure, which results in an edge in the constraint network that relates the current robot pose with a former robot pose. These loop closure edges contradict the pose estimates resulting from plain odometry. In order to correct the structure of the graph after a loop is detected, the graph optimisation algorithm TORO (Grisetti et al., 2007b) is used, which considers each edge in the pose graph as a cost function for each two connected nodes, and rearranges the nodes in the graph such that the total costs associated with all edges are minimised. Fig. 4.2 shows an overview of the mapping system.

4.2.1 Relations Based on Odometry

Let the robot's pose at any given time step t be represented as $\mathbf{p}_t = (x_t, y_t, \theta_t)$, where (x_t, y_t) are the coordinates of the robot and θ_t the current heading. In the stop-sense-go strategy, robotic motion between two consecutive poses is approximated by translation d followed by rotation δ, and the model which obtains the

pose $\mathbf{p}_t$ from $\mathbf{p}_{t-1}$ is

$$\begin{pmatrix} x_t \\ y_t \\ \theta_t \end{pmatrix} = \begin{pmatrix} x_{t-1} \\ y_{t-1} \\ \theta_{t-1} \end{pmatrix} + \begin{pmatrix} \hat{d}_t \cos(\theta_{t-1}) \\ \hat{d}_t \sin(\theta_{t-1}) \\ \hat{\delta}_t \end{pmatrix}, \tag{4.1}$$

where $\hat{d}_t = d_t + \epsilon_d$ and $\hat{\delta}_t = \delta_t + \epsilon_{rot}$ are obtained from the odometry measurements by adding independent Gaussian noise, where

$$\epsilon_d \sim N(0, \omega_{range}|d|), \tag{4.2}$$

$$\epsilon_{rot} \sim N(0, \omega_{turn}|\delta_{turn}| + \omega_{drift}|d|), \tag{4.3}$$

where ω_{range}, ω_{turn} and ω_{drift} represent the range error, the turn error and the drift error of the robot encoder, respectively. (Fig. 4.3 illustrates the motion of the robot between two consecutive poses.)

4.2.2 Tracking of the robot heading

As mentioned above, the translation and rotation movements of the robot between two consecutive poses are measured from the odometry, which is noisy and prone to error. Taking into account that the robot is performing a short translation followed by a rotation between each consecutive pose, the majority of the error in measuring the movement of the robot originates from rotation due to friction between the wheels and the floor when the robot is rotating.

In order to reduce errors involved in measuring the movement of the robot

as much as possible, an extended Kalman filter (EKF) (Gordon et al., 2002) is used. In general, the extended Kalman filter is used to track the pose of a robot, i.e. the position and the heading orientation, by using the motion model of the robot to predict how the pose should be changed according to odometry measurements, and then utilising sensory observation to correct its estimation. However, in our case, only the heading component of the robot pose is observed using vision, taking into account that when a loop is detected in the map, the graph optimisation algorithm (TORO) will correct its overall geometry.

The tracking is done as follows: at step t, the relative orientation between the current pose of the robot $\mathbf{p}_t$ and the previous pose $\mathbf{p}_{t-1}$ is computed from the vision sensor using the two omnidirectional images that were recorded at each position. Then, by adding this relative orientation to the heading from the previous step, θ_{t-1}, a vision-based observation for the robot's heading is obtained at step t. The next step is feeding this heading observation into the EKF filter, which uses the robot motion model from Eq. 4.1 as a prediction step and estimates the robot's pose at step t.

4.2.3 Loop Closure Using Vision

A loop-closing capability is an important part of any mapping system because, without this capability, the robot can face the problem of assigning the same area in the environment to multiple nodes, thus leading to a globally inconsistent map. Therefore, the robot uses its vision sensor to detect loops along the trajectory by using place recognition based on image similarity. As mentioned earlier, each node in the map contains a group of image feature points extracted from an

omnidirectional image recorded when the node was first created. Using these image features, the similarity between any two nodes in the map is measured using the number of matched feature points between the two groups of features stored in each node. So, in order to detect loop closures, the robot calculates the similarity between the current node and all the nodes located within a certain radius. When the ratio of the number of matched feature points to the number of features stored in the current node exceeds a certain threshold, a loop is considered detected. As a result, the robot adds a link between the two detected nodes in the graph with a distance of zero. Subsequently, TORO, the graph optimiser, is run on the graph to correct the pose estimates of the nodes in the map according to the newly added link. TORO accepts as an input a directed graph $G = (V; E)$ with nodes V and edges E where each node X_i in V represents a robot pose, and each directed edge (i, j) in E represents a transformation $T_{i,j}$ that takes pose X_i to pose X_j. The optimisation step aims to select the spatial configuration of the nodes, which best satisfies the constraints encoded in the edges.

Two important parameters appear in the above loop closure strategy. First, there is the radius in which the robot checks for a loop closure whereby the longer the radius, the more nodes need to be matched with the current node. In the experiments, a pre-selected radius of 3 m, measured as an Euclidean distance between the nodes, was used; however, one can also use the Mahalanobis distance to obtain a more accurate representation for the distance between the nodes, based on uncertainty in the estimation of the robot's position.

Second, we can consider the matching threshold that the robot uses to decide that a loop has been closed. This threshold is affected by the texture of the mapped environment, which in turn affects the number of image features that

can be extracted from the images. In the experiment, a loop is considered closed if 35% of the features are matched.

4.3 Graph Pruning

The resulting map from the above step is a graph containing nodes created at each step performed by the robot; consequently, the graph is dense and contains redundant information which can be removed. This section describes the extraction of a sparse topological map from the dense graph by deploying a graph-pruning algorithm to reduce the number of nodes.

A naive solution to reduce the number of nodes in the graph would be based on the time stamp of the nodes, where the graph is sampled using a fixed time step. This method would fail if the robot stood still for some time or changed speed while mapping the environment. Another simple solution would be based on the distance between the nodes (Jogan and Leonardis, 2003). However, this method does not take into account the rapid change in the appearance of the robot's surroundings, which could occur when the robot crosses a doorway or goes round a corner. Successive images can differ considerably on the two sides of the doors or corners, which could become a challenge when the robot tries to actually use the map for navigation. This implies that image similarity should be considered in the process.

In the field of video analysis, one can find similar approaches where a set of key frames are selected to give a visual summary of a long video. These methods are used to generate a so-called "navigation summary", where a set of a small number of key frames is selected to summarise the visual experience of a mobile

robot. The robot can summarise, for example, a video recorded from a long trip or a video for monitoring patients in an intensive care unit. Recently, methods based on Bayesian surprise (Itti and Baldi, 2009) have become popular (Girdhar and Dudek, 2010), as they can detect when measurements have high unlikelihood according to the current model of the environment. Therefore, in the the case of selecting a set of images to summarise a long sequence, the images which represent a sudden change in the appearance will be selected. This method can be used in our case to prune the dense graph, where key images can be selected in locations such as the entrance to a room or the intersection of two corridors (Ranganathan and Dellaert, 2009). However, due to the nature of the summarising process, which concentrates on selecting only those images which reflect sudden changes in the appearance of the environment, the number of selected images will be small and the generated map too sparse. This can be a problem in applications where the robot needs to use the map for visual navigation, and arises because there will be long distances between the nodes, and the robot will have no directional clues in the gaps between the nodes.

Similarly, for mobile robotic applications, methods that use image similarity to reduce the number of nodes in a dense topological map start by clustering the nodes based on image similarity, and then choose key nodes as a representative for each image group (Booij et al., 2008). These methods can be used when information about the geometric distances between the nodes is not available, although in our case the distances between the nodes are provided and it is possible to use them as an additional source of information. Instead of choosing the key nodes in the graph based on the geometric distance or the image similarity alone, we propose a method that selects the nodes in the graph based on a

combination of the two metrics.

Recently, Ila et al. (2010) introduced a pose graph mapping method whereby they measure statistical content on links to enforce node sparsity and limited graph connectivity. The result is a compact map which contains non-redundant nodes and links. The main criteria with which the nodes are selected relate directly to reducing uncertainty in the estimation of the robot's position. This tends to produce densely distributed nodes when the robot performs curved paths, and sparsely distributed nodes when the robot performs straightforward paths, because of the increasing uncertainty resulting from turning motions. The method presented in the next section uses different criteria to reduce the number of nodes in the map. The main goal is to produce a map with sparse nodes located in places that are suitable for using the map for visual navigation tasks.

4.3.1 The Dual Clustering Algorithm

Let us consider the initial dense graph map as a set of spatial objects, with the nodes representing these objects. Each node has two attribute domains, a geometric domain in the xy-plane and a non-geometric domain represented by image similarity. The aim is to select a subset of these nodes in a way that sufficiently covers the environment, allowing the robot to use the map for tasks such as autonomous visual navigation. In order to do that, a clustering method called "dual clustering" (Lin et al., 2005) is used. Dual clustering is the process that partitions a set of spatial objects into different clusters using two attribute domains, geometric and non- geometric, in such a way that each cluster forms a compact region in the geometric domain (the distance between the nodes in our

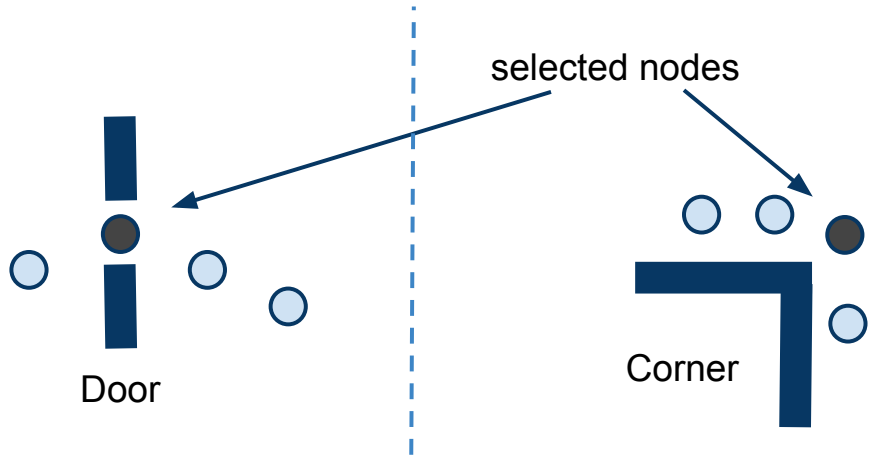

Figure 4.4: Maximising intra-cluster similarity aims to automate the selection of reference views from areas such as the middle of the doorways and the edge of the corners.

case) while maximising the intra-cluster similarity in the non-geometric domain (the image similarity in our case).

In the geometric domain, the clustering algorithm produces compact clusters, which is a preferable effect because the aim is to avoid the selected nodes, which correspond to the centres of the clusters, being very sparse, creating big gaps in the final map. In the non-geometric domain (i.e. image similarity), the clustering algorithm selects the centres of the clusters in a way which maximises intra-cluster similarity. The effect of this process is also preferable for our case because, in the cases where a cluster of nodes from the initial dense graph expands through a doorway or a corner, the centre of the cluster will be selected from the middle of the doorway or the edge of the corner. This selection allows the node to cover both sides of the door and the corner, preventing discontinuity in covering the environment, which can be a problem when actually using the map for tasks such as visual navigation. Fig 4.4 illustrates this situation.

Inspired by the fast implementation of the dual clustering method presented

in (Zhou et al., 2007), clustering is performed based on density in the geometric domain, while maximising the intra-cluster similarity in the non-geometric domain. For each cluster centre in the geometrical domain, the neighbourhood of a given radius Ψ_d should contain a minimum number of points, while in the non-geometric domain the similarity between the neighbourhood points of each cluster and the centre of that cluster should be above a certain threshold Ψ_s. The clustering process is performed incrementally as follows:

1. Initialise a cluster by starting from the first unclassified node in the graph N_p (based on the time stamp). If the number of nodes in the neighbourhood of N_p is less than a predefined threshold κ, the node is ignored. Otherwise, create a new cluster C.

2. Insert all nodes from the neighbourhood of N_p, which have similarity with N_p greater than Ψ_s, into C.

3. Compute the centre of the cluster N_{ref} by selecting the node which is most similar to all other nodes in the cluster, as

$$N_{ref} = \arg\max_{k \in C}(\sum_{j \in C} sim(N_j, N_k)),$$

   where $sim(N_j, N_k)$ is the similarity between the two views in nodes N_j and N_k as the number of matched image features.

4. Check an arbitrary node N_q, from the neighbourhood of N_p. If the number of nodes in the neighbourhood of N_q is at least κ, and the similarity between an unclassified node N_o in the neighbourhood of N_q and the centre of the cluster is above Ψ_s, then insert N_o into cluster C and recompute the centre

of the cluster as in step 3. Repeat step 4 until the cluster C cannot be extended any more.

5. Repeat all the steps until all the nodes in the graph are classified.

A discussion about how to set the clustering parameters for the experiments is presented in Section 4.5.

4.4 Using the Map for Navigation

Every map can be judged by its usefulness for practical purposes. In our case the map is used for a path-following routine inside an indoor environment.

When robots work inside an indoor environment, their navigation generally is restricted to what humans consider to be a path inside that environment, such as corridors and areas between furniture. These routes effectively simplify the task of navigation by limiting the robot to only one degree of freedom along the path. By representing this path as a sequence of images, the following framework for appearance-based navigation is generally used in the literature (Blanc et al., 2006; Booij et al., 2007; Goedemé et al., 2007; Matsumoto et al., 2002; Ohno et al., 2002):

- The path is first built during a learning phase in which the robot is controlled by a human operator. During this phase the robot captures a sequence of images along the path.

- A subset of captured images is selected to represent reference images along the path.

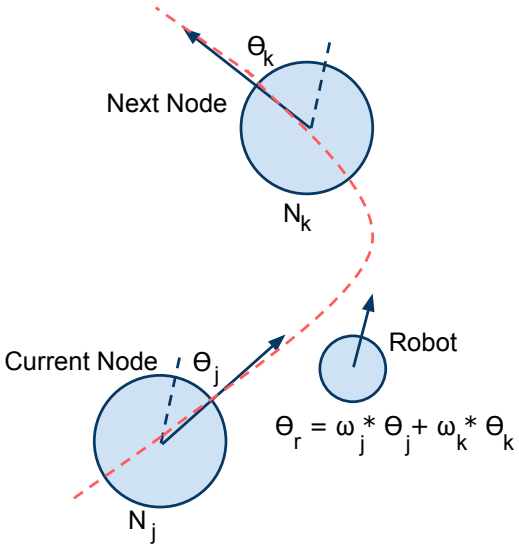

Figure 4.5: The proposed visual navigation strategy. N_j is the current node in the path and N_k is the next node. θ_j and θ_k are the relative orientations between the robot's heading and the reference orientation of the nodes N_j and N_k, respectively. θ_r is the robot's desired heading.

- During the replay phase, the robot starts near the first position and is required to repeat the same path.

- The robot extracts the control commands of its motion by comparing the currently observed image with reference images along the path.

This work adopts a similar framework for visual path-following using a sequence of nodes from the map. Fig. 4.5 illustrates the navigation strategy. First, the robot localises itself to one of the nodes in the path, which is done by selecting the node that has the highest similarity score with the currently observed view. Other localisation approaches, e.g. Markov's localisation (Fox, 1998), could also be used to improve performance robustness.

Let S_j and S_k be the similarity score between the robot's current view and

the views stored in the nodes N_j and N_k, respectively, then the following weights are computed as

$$\omega_j = \frac{S_j}{S_j + S_k}, \ \omega_k = \frac{S_k}{S_j + S_k}. \tag{4.4}$$

Finally, the desired heading angle of the robot, θ_r, is computed as a weighted sum:

$$\theta_r = \omega_j * \theta_j + \omega_k * \theta_k, \tag{4.5}$$

where θ_j and θ_k are the relative orientations between the current view and nodes N_j and N_k, respectively (see Fig. 4.5). By following this navigation strategy, the nodes in the path can be considered as directional signs, which lead the robot toward its goal.

In order to estimate the relative orientation between two views, epipolar geometry is used. As presented in Section 3.3.2.1, first the essential matrix $\mathbf{E}$ is estimated. Then, based on the method introduced by Hartley and Zisserman (2004), it is factored to give Eq. 4.6, which contains the rotation matrix $\mathbf{R} \in SO(3)$ and the skew-symmetric matrix $[\mathbf{t}]_\times$ of the translation vector $\mathbf{t} \in \mathbb{R}^3$.

$$\mathbf{E} = [\mathbf{t}]_\times \mathbf{R}. \tag{4.6}$$

Then the relative orientation θ is extracted from the rotation matrix $\mathbf{R}$:

$$\mathbf{R} = \begin{bmatrix} \cos(\theta) & \sin(\theta) & 0 \\ -\sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{4.7}$$

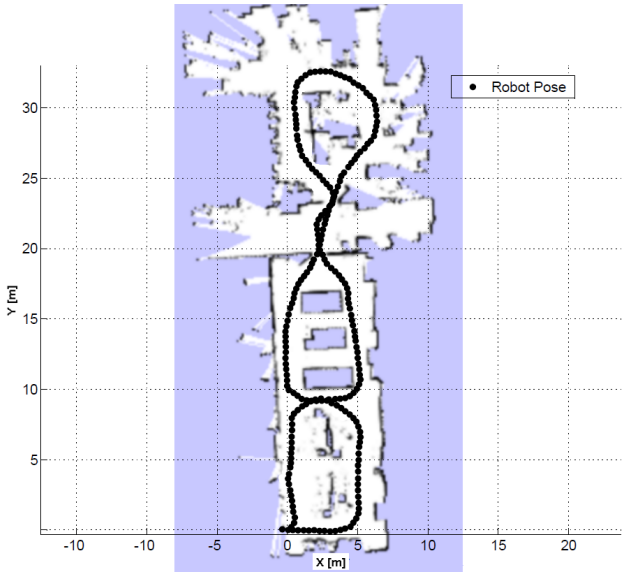

Figure 4.6: The true trajectory of the robot obtained using a laser-based SLAM algorithm.

4.5 Experimental Evaluation

The following experiment was carried out on an office floor at the University of Lincoln, where the robot was driven on a tour between the offices while recording a set of omnidirectional images. The resulting database contains 222 images, with approximately 35 cm between each consecutive images. Fig. 4.6 shows the positions of the recorded images inside an occupancy grid map of the environment, which was obtained using a laser-based SLAM library (Grisetti et al., 2007a). Information obtained from the laser-based SLAM was used for visualisation purposes only.

The robot started the mapping process from location $(x_0 = 0, y_0 = 0)$ and finished by coming back to the same start point. The dashed line in Fig. 4.7 shows the trajectory of the robot based on odometry alone, where the drift effect

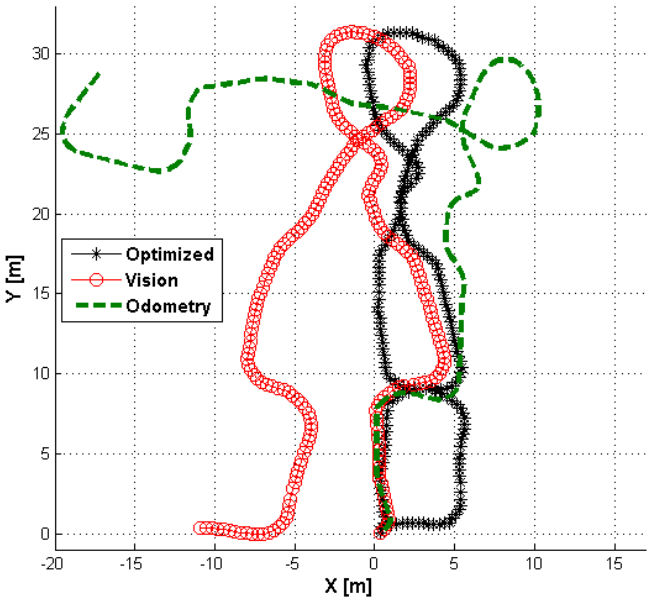

Figure 4.7: The green dashed line represents the trajectory of the robot from the odometry. The red line is the result of using vision estimated relative orientation as an observation with an EKF filter. The black line is the final output after loop closing and graph relaxation.

is clear. Fig. 4.7 also shows the trajectory of the robot with the EKF as a filter for observing the robot's heading. Although the robot did not have any method for detecting loop closure in that sequence, the resulting error in estimating the trajectory was smaller. The range error ω_{range} used for Eq. 4.2 was 0.005 m, and the turn error ω_{turn} was 3.0^o and the drift error ω_{drift} was 0.5^o. The motion noise was computed by moving the robot on a marked ground and measuring the error in the motion from the odometry compared to the groundtruth. The observation noise was computed by turning the robot by hand for a known angle and then estimate the same rotation angle from vision and compare the error between both.

The final output of the mapping step is also shown in Fig. 4.7, which highlights

that the graph optimisation algorithm was invoked each time a loop was detected in the graph. The final map is a dense graph with nodes created at every step the robot has performed; therefore, a graph-pruning process is applied in the next step to remove redundant nodes from the map.

4.5.1 Graph Pruning Results

the number of nodes in the final sparse map is affected by the neighbourhood radius Ψ_d and the image similarity threshold Ψ_s. Parameter Ψ_d gives an initial radius for the node to cover, and then the algorithm expands the node based on the image similarity threshold Ψ_s. If the map is intended to be used for autonomous visual navigation, its sparsity should not result in gaps where the robot cannot estimate its heading relative to one of the nodes in the map. In order to achieve that, parameter Ψ_d was assigned to 1 m as a minimum distance between the nodes. The parameter Ψ_d is the initial minimum distance between the nodes and by increasing its value the number of images which needs to be compared to the node's centre will increase which leads to producing fewer nodes in the final map.

The image similarity threshold Ψ_s was assigned to 35 feature points as the minimum number of matched points between any view in the node and the centre of the node. The values of 35 feature points was selected after a manual tuning for the method which estimate the essential matrix between two spherical views, which shows that at least 35 image features were required for the estimation of the essential matrix otherwise a degenerate solution could be obtained. On the other hand, increasing the value of Ψ_s will result in making the nodes cover

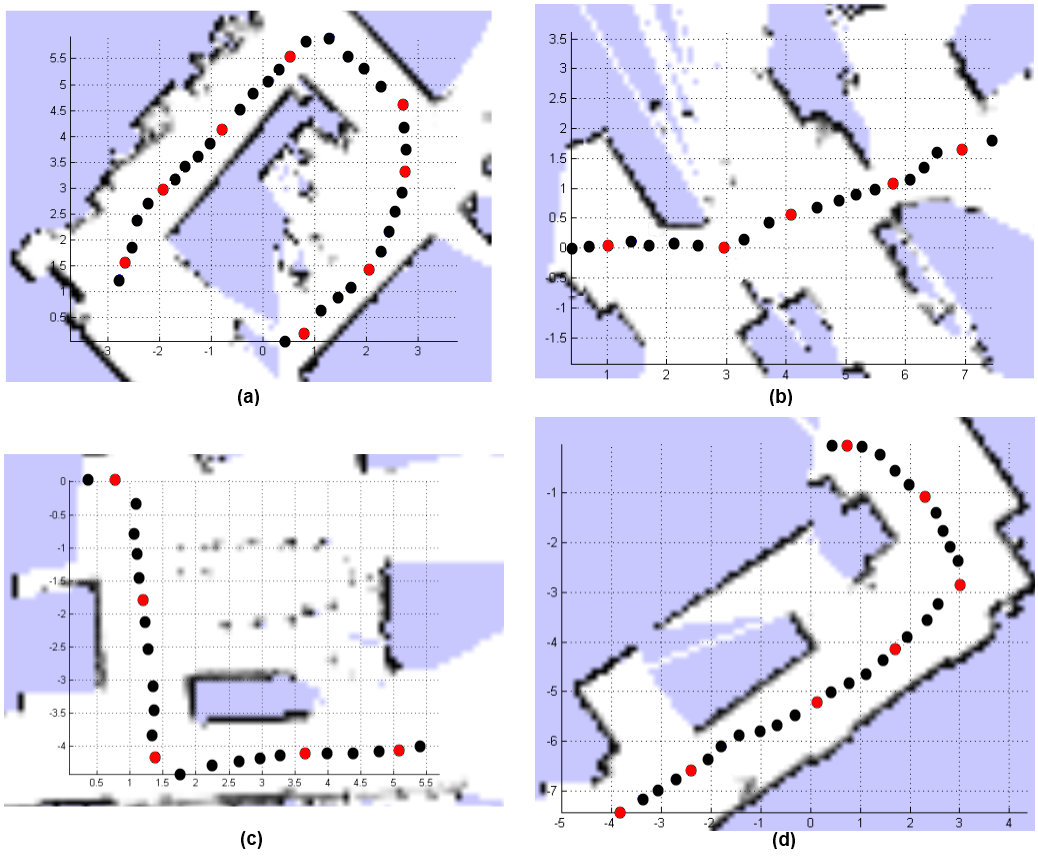

Figure 4.8: Four short image sequences recorded from different parts of the environment that include a corner or a doorway. The black points are the positions of the images and the red points are the selected nodes allocated by the dual clustering algorithm.

smaller areas, which in turn leads to increasing the number of nodes in the final maps.

Finally, parameter κ was assigned to 1 as the minimum number of points required in the neighbourhood of any node using the assumption that the robot perform a continuous movement while capturing a sequence of images.

4.5.1.1 Corners and Doorways

In order to show the behaviour of the algorithm when the robot goes through corners and doorways, four short sequences of images were recorded from different parts of the environment that include a corner or a doorway. Dual clustering was then applied to select the nodes in these sequences. Fig 4.8 shows four laser occupancy grids for the sequences, with the selected nodes marked in red. Fig 4.8(a) shows an occupancy map for a sequence of corners and indicates that the nodes are selected from the edges of the corners. Fig 4.8(b) shows an occupancy map for two rooms connected by a corridor. Again, the map shows that two nodes are selected from the middle of the two doorways. Fig 4.8(c,d) are two single corners. The occupancy map shows that there are nodes selected from the edge of each corner, as expected.

4.5.1.2 The Final Map

In this step, the dual clustering algorithm was applied, which pruned the dense graph produced from the previous step and generated the final map.

In order to compare the output of the dual clustering algorithm with other methods, such as selecting nodes based only on the distance between them or on image similarity, the pruning process was repeated three times. On the first occasion, dual clustering was used. The second time, only the distances between the nodes were used to select the nodes, while at the third attempt only image similarities between the nodes were used to select the nodes. Fig. 4.9(a) shows the result of the pruning step, where a set of 23 nodes was selected using the dual clustering algorithm. Fig. 4.9(b) shows the result of selecting the nodes based

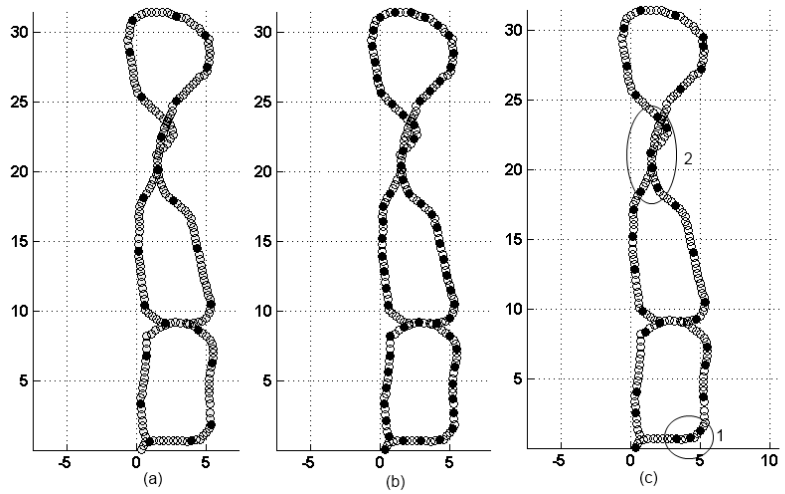

Figure 4.9: (a) shows the 23 selected nodes based on the dual clustering algorithm, with a 1 m distance threshold and 35 image features for the similarity threshold. (b) shows the 62 selected nodes when only a threshold of 1 m is used. (c) shows the 34 selected nodes when only a threshold of 35 image features is used.

only on the distance between the nodes. Using a 1 m distance threshold results in 62 uniformly distributed nodes over the trajectory of the robot. Compared to the dual clustering method, when using the same value of 1 m for the distance threshold, it is possible to note that using only the distance results in three time more nodes in the map, without any consideration for the positions of the selected nodes related to the appearance of the environment. On the other hand, Fig. 4.9(c) shows the result of selecting the nodes based only on the similarity between views in the map. This was achieved as follows. Each image in the sequence is compared to all the nodes' centres. If the similarity between the image and each one of the centres is below similarity threshold Ψ_s, the image is added to the map as a new node centre; otherwise, it is assigned to the node with the highest similarity score. The result is the selection of 34 nodes using the same similarity threshold of 35 feature points, which was used for the dual

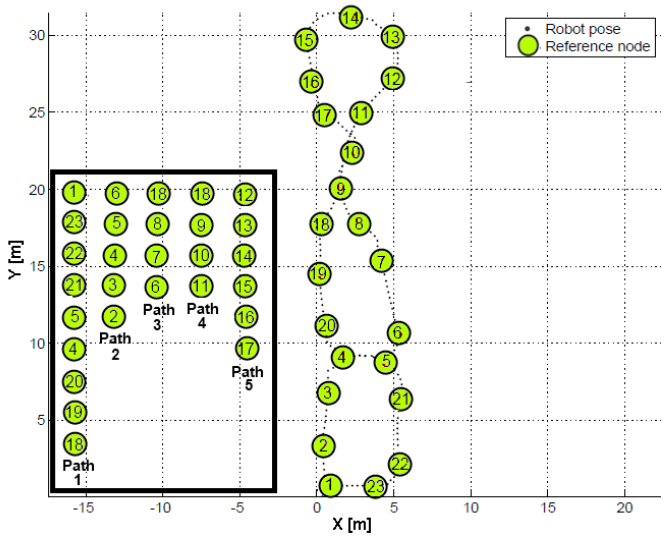

Figure 4.10: Selected nodes from the dual clustering algorithm, along with five node sequences the robot was given to follow.

clustering algorithm. Compared to the results from the dual clustering method, it is noteworthy that the nodes tend to concentrate in areas where there is a rapid change in appearance. One example of such a situation appears in area 1 and 2 in Fig. 4.9(c). This results in using more nodes to cover the corners and the doorways between the rooms.

4.5.2 Visual navigation performance

In order to test the map for visual navigation, five path-following runs were performed using the map nodes. The paths were chosen randomly, while covering all nodes in the map (see Fig. 4.10). At the start of each run the robot was given a sequence of nodes to follow, which was achieved by using the navigation strategy presented in Section 4.4 and employing an array of sonar sensors for obstacle avoidance.

The obstacle avoidance procedure used in this work is as follows. When the robot receives a command to rotate, which is based on its current view and the views from the nodes in the map, it checks the sonar range readings first. If the sonar ranges allow for movement, the robot simply executes the movement; otherwise, it turns 10^o in the opposite direction and then moves forward for a distance of 50 mm. After that it re-estimates the desired heading using the view from its new position. If both directions are blocked, the robot moves backward 100 mm and then re-estimates the desired heading from its new position. Finally, if the robot receives a command to move forward but there is no room for the movement based on sonar readings, the robot checks the sonar ranges on its right and left sides and then turns in the direction which has the most free space.

The same five runs were also re-executed using manual drive. A human driver steered the robot, taking the best track where the robot was driven through the shortest distance and at the same time was kept away from obstacles. Table 4.1 shows the results for both the autonomous and manual runs.

In order to show the robustness of the navigation procedure, 5 autonomous runs were executed twice. The mean and minimum sonar range distances to obstacles were calculated for each run, along with the travelled distance. These values were used as an indication about the quality of the navigation performance. If the navigation performance was poor, the robot would be steered towards obstacles and away from the goal, which leads to short sonar range distances to obstacles. As the results show, although the robot took a slightly longer distance to reach its goal, the autonomous routes were smooth and similar to those recorded for the manual runs. The average distance to any obstacle was 1.00 m in the case of the manual runs, whereas the average for the autonomous

runs was 0.88 m. The average value of the minimum range to obstacles was 0.49 m for manual driving and 0.44 m for autonomous operation.

The autonomous navigation was repeated only twice for this experiment just to give the reader a sense about the performance, however later on experiment 6.5, the same autonomous navigation strategy is repeated 38 times over a period of 7 weeks and the performance was analysed in more details.

Table 4.1: The results from five autonomous path-following runs executed twice, Auto1 and Auto2. The paths were chosen randomly, while covering all nodes in the map. The same five runs were also re-executed using manual drive (Manual). A human driver steered the robot, taking the best track where the robot was driven through the shortest distance and at the same time was kept away from obstacles. The results shows the distance traveled by the robot for each run, the mean sonar range to obstacles during each run and the minimum sonar range during each run.

		Distance [m]	Mean Range [m]	Minimum Range [m]
Path 1	Manual	22.87	0.85	0.43
	Auto1	24.07	0.82	0.42
	Auto2	24.28	0.82	0.44
Path 2	Manual	14.30	1.20	0.58
	Auto1	15.34	0.90	0.45
	Auto2	14.90	0.99	0.48
Path 3	Manual	9.81	0.94	0.56
	Auto1	10.66	0.85	0.41
	Auto2	10.70	0.87	0.51
Path 4	Manual	8.97	1.04	0.42
	Auto1	9.27	0.97	0.37
	Auto2	9.09	1.02	0.42
Path 5	Manual	15.52	0.98	0.48
	Auto1	16.42	0.85	0.46
	Auto2	16.69	0.78	0.47

4.6 Summary

This chapter presented a minimalistic mapping method for a mobile robot using an omnidirectional vision sensor. The produced map is hybrid with two levels of representation, namely global and local. On the global level, the world is represented as a graph of adjacent nodes, with each node containing a group of image features used to represent the appearance of the node. On the local level, the features inside each node form a spherical view, which is used for visual navigation. The map is built using a sequence of images along with odometry information. The global consistency of the map is achieved using a graph optimisation algorithm. In order to reduce the number of nodes in the map, a dual clustering algorithm for post-processing the initial map was developed. The map was used in an experiment where the robot performed multiple path-following tasks.

The next chapter presents an updating mechanism employed by the robot to update the map over time. The hybrid map which the robot built using the method presented in this chapter is used for a long-term operation over a relatively long period of time. The updating mechanism, which will be introduced next, works on the spherical views of the map by adding and removing image features from the nodes while maintaining the ability to use the map for navigation using the navigation method presented earlier in Section 4.4.

Chapter 5

Long-Term Map Updating

This chapter first presents a long-term updating mechanism using a multi-store memory model that is inspired by the basic theory about the structure and functionality of the human memory. Next, the long-term updating mechanism is extended to incorporate a spherical view representation of image features stored in the nodes of a hybrid map. Finally, the results from an experimental evaluation of the stability of the system, and the effects of choosing different values for the parameters of the system, are presented.

5.1 Introduction

Similar to a situation when new members of staff are given a tour around their workplace to help them build a mental map of their new surroundings, mobile robots could use their first tour through their working environment to build a map that contains sensory observations about appearance and/or geometry of the environment. However, unlike the case with the new members of staff, who have

the ability to adapt and change their mental map effortlessly as the appearance of their working environment changes over time, mobile robots need to have a mechanism to update the initial map, which has been built from the first tour; otherwise, the map will eventually become out-of-date and unusable.

Following the above analogy, it is assumed that a mobile robot is first introduced to its workplace, during which time it builds a hybrid map of the environment using the method presented in Chapter 4. After this initial step the robot uses an updating mechanism, which is presented in this chapter, to continually update the map over time. The updating mechanism involves a long-term learning and forgetting process which adds and removes image features stored in the spherical views of the hybrid map in response to changes in the appearance of the environment. The updating mechanism is inspired by the multi-store model of human memory proposed by Atkinson and Shiffrin (1968), which allows the robot to detect when a change occurs to the appearance of the environment and also provides a mechanism to select only stable sensory observations to become part of the map.

5.2 Biologically Inspired AI Approaches

Adopting biologically inspired approaches to build and operate mobile robots that have to work among us and inside our everyday environments sounds very appealing because they will work in the same context in which we, as biological systems, operate. As a result, they will face the same challenges we face in the course of our daily life. One of the most important challenges mobile robots need to overcome in this respect is the ability to adapt to various changes in

their working space, from changes in the appearance of a room to changes in the behaviour of people.

From a biological point of view, an organism is adaptable if it can survive significant changes to its environment, spread to new habitats and come up with novel solutions that will allow it to survive in new surroundings. For humans, adaptability is a key factor for survival and depends heavily on accurate evaluation of environmental changes. This accuracy is a product of our personal experience, which in turn is collected and stored in our memory.

The human memory is not yet fully understood, and many of the ideas about how it works are still controversial. One idea that is widely accepted is to look at the memory in terms of multiple stores of information and a set of processes that act on these stores. Therefore, in this work, the multi-store theory is adopted and brought to the domain of mobile robot mapping and localisation. The theory is used to enable the robot to adapt its internal representation of the world over time in response to the changing appearance of its surroundings.

In the broader domain of artificial intelligence, one can find approaches that are inspired by the multi-store model of the human memory. For example, Peters (2010) proposed an object-based memory based on the multi-store model for virtual agents equipped with a synthetic vision system. Vargas et al. (2010) proposed hierarchical classification data mining techniques to implement forgetting and memory generalisation mechanisms in robotic companions. Mavridis and Petychakis (2010) identified a list of 16 desiderata for human-like memory systems in socially interactive robots. These approaches focus mainly on improving the human-robot interaction process, whereas in this work the focus is on using the memory model for spatial representation.

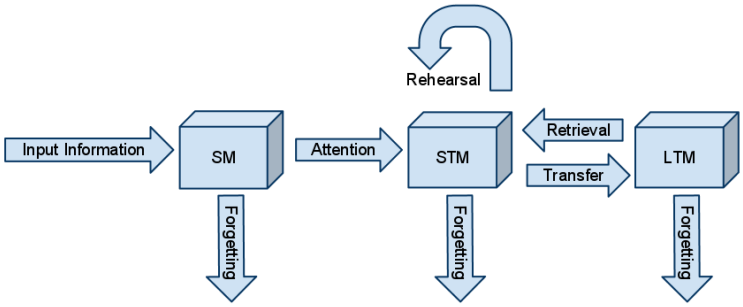

Figure 5.1: The multi-store model of human memory in its simplest form.

In the following sections, a brief overview of the general structure and functionality of the multi-store model of human memory is presented, followed by a description of the updating mechanism for long-term mapping using the concepts inspired by the biological system.

5.3 The Multi-Store Model of Human Memory

According to the basic model of Atkinson and Shiffrin (1968), which forms the basis of most modern memory theories, human memory is divided into separate stores – sensory memory (SM), short-term memory (STM) and long-term memory (LTM). Figure 5.1 presents the three stores of memory and the interactions between them.

- **Sensory Memory (SM):** The sensory store is the first stage where information is stored. The word "sensory" is a reference indicating that the information received in this store is sensual (e.g. visual, auditory, olfactory, tactile). The capacity of this store is believed to be vast, but the duration of storing the information is only 0.25 to 2 seconds (Baddeley et al., 2009). Selective attention, which involves the long-term memory, determines what

information moves from sensory memory to short-term memory (e.g. taking notice of something familiar or alarming).

- **Short-Term Memory (STM):** Short-term memory is the smallest store in the model. Its role is to provide the ability to keep a small amount of information in an active, readily available state for a short period of time. The widely accepted theory about the capacity of the STM store is 7 ± 2 pieces of information such as names of objects and phone numbers. The duration of keeping the information in this memory is believed to be in the order of seconds, and the encoding of the information is mainly visual and acoustic. Through the process of rehearsal, information in STM can be committed to LTM, and then be retained for longer periods of time.

- **Long-Term Memory (LTM):** Long-term memory is a large store that holds information for a very long time. The LTM store affects our perception of the world and influences what information we attend to in our surroundings. The information from this store is retrieved using a process called recall. According to Tulving's model (Tulving, 1972), LTM can be divided into declarative and procedural memory; declarative memory is further divided into semantic and episodic memory, the latter of which provides the capacity to remember specific events, while semantic memory stores accumulative knowledge of the world (e.g. some generalised representation of different episodes experienced).

A number of the assumptions underlying this basic model have been subsequently questioned, causing the model to be further elaborated (Baddeley et al., 2009).

5.4 Multi-Store Memory for Mobile Robots

While robotic memory need not to be constrained by the fallibilities of human memory, nor by the exact details of its biological implementation, the multi-store model of human memory provides a natural framework for the filtering and storage of perceptual information in artificial agents such as robots.

As presented in Chapter 4, the robot's world in our case is represented as a graph of nodes corresponding to places in the real environment. Each node in turn is represented as a set of image features that describe the appearance of the environment. A spherical view representation is used for each node, where the image features are projected onto a unit sphere. This representation enables the robot to use features for visual navigation.

Applying the concepts of the multi-store model of the human memory to robot mapping, sensory memory will contain image features extracted from the current image of the robot's camera. Then, an attentional mechanism selects which information to move to STM, which is used as an intermediate store where new observations are kept for a short time. Over this time the system uses a rehearsal mechanism to select features that are more stable for transfer to LTM. In order to limit the overall storage requirements and adapt to changes in the environment, the system also contains a recall mechanism that forgets unused feature points in LTM by removing them from the given node. LTM is used in turn by the attentional mechanism for selecting the new sensory information to update the map. Thus, only *changes* to the mapped environment are selected for input to STM.

5.4.1 Recall, Rehearsal and Transfer

Each node in the map has two memory stores: STM, which is a temporary stage, and LTM, which provides reference views in the map used for self-localisation. Assuming that the robot is able to self-localise by matching features extracted from the current view to the stored reference views, the purpose of the algorithm presented here is to maintain up-to-date reference views for the nodes in the map, using recall and rehearsal concepts inspired by human memory.

To initialise the LTM and STM stores of the map, the image data of the environment from the robot's first tour is used. For each node, the group of image features, which are extracted from an omnidirectional image captured in the place corresponding to the node, are used directly to initialise LTM, while STM for each node is initially assigned as empty.

Thereafter, every time the robot visits an existing node, the following steps are carried out. Feature points are extracted from the current view. Self-localisation is carried out by comparing the current features to the reference features of each node (LTM), in order to identify the current node. In our case, global localisation is applied by place recognition using a simple winner-takes-all strategy, although any appropriate self-localisation algorithm could be applied, e.g. Markov localisation. After localisation, current features are used in the recall stage for updating the LTM of the current node. Only new features which do not match any of the features in LTM are used in the rehearsal stage. Algorithm 1 describes the two main stages: recall, where the difference in appearance between the reference and current view is computed, and rehearsal, where this difference is used to update STM and commit persistent new features from the view of the node to LTM.

Algorithm 1 Reference view update procedure.

Definitions:
CrrNode: The reference view of the current node.
CrrView: The current view for the current node.
CrrSTM: The current STM for the current node.
STMlng: The maximum number of states in the STM.
LTMlng: The maximum number of states in the LTM.
newFP: The difference between the CrrView and CrrNode.

---

for (every visit to the node) {
    newFP = **recall(** CrrNode , CrrView , LTMlng **)**
    rehearse(CrrSTM, newFP , STMlng **)**
}

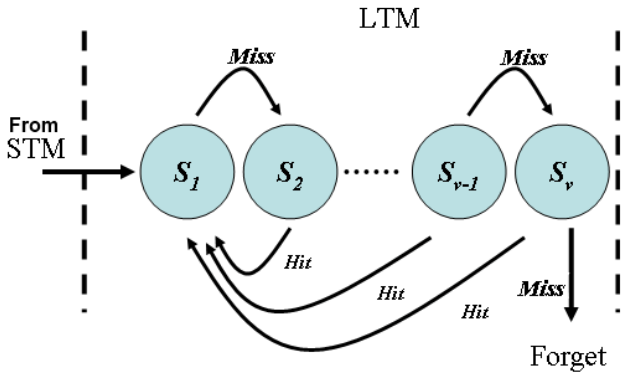

Figure 5.2: The recall procedure in LTM.

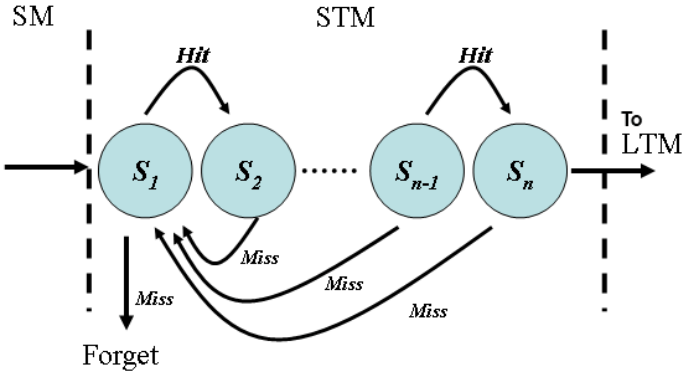

Figure 5.3: The rehearsal stage in STM.

Algorithm 2 shows the recall process for a feature stored in LTM, which is also represented as a finite state machine in Fig. 5.2. This process first involves

Algorithm 2 The recall stage in LTM.

```
newFP = [ ]
for (every feature in CrrNode) {
    if (feature in CrrView){
       Reset the feature to the first state.
    else
       Move the feature to the next state.
    }
    if (feature state > LTMlng) {
       Remove the feature from CrrNode.
    }
}
for (every feature in CrrView) {
    if (feature not in CrrNode) {
       Add the feature to newFP.
    }
}
return newFP
```

Algorithm 3 The rehearsal procedure in STM.

```
for (every feature in CrrSTM ) {
    if (feature in newFP) {
       Move the feature to the next state.
       if (feature state > STMlng){
          Move the feature to the CrrNode.
          Remove the feature from CrrSTM.
       }
       else if (feature in the first state) {
                Remove the feature from CrrSTM.
       }
       else {
                Reset the feature to the first state.
       }
    }
}
for (every feature in newFP ) {
    if (feature was not in CrrSTM) {
       Add the feature to CrrSTM in the first state.
    }
}
```

updating the LTM by matching the reference view to the current view. In order to remain in LTM, a feature has to be seen occasionally in that node. In contrast

to rehearsal, features enter LTM from STM and must progress through several intermediate states (S_1 to S_v) before being forgotten. Stored features which have been seen in the current view are reset to the first state (S_1), while the state of features which have not been seen is progressed and a feature that passes through all states without a "hit" is forgotten. Finally, recall returns the list of new features that were not already present in LTM (i.e. the difference in appearance between the current and reference views).

Algorithm 3 shows the rehearsal process for a stored feature in STM, which is also represented as a finite state machine in Fig. 5.3. This stage represents what Atkinson and Schiffrin called "rehearsal" in their memory model (see Fig. 5.1), i.e. the process of continually recalling information into STM in order to memorise it. In order to transfer a feature point from STM to LTM the feature has to be frequently seen in that node. Features enter STM from sensory memory and must progress through several intermediate states (S_1 to S_n) before their transfer to LTM. Every time the robot visits a node and finds the feature ("hit"), the state of the feature is moved closer to LTM. However, if the feature is missing from the current view ("miss"), it is returned to the first state (S_1) or forgotten if it is already there. This policy means that spurious features should be quickly forgotten, while only persistent features will be transferred to LTM.

5.4.2 Updating the Spherical Views

Updating the reference views of the map based on the proposed memory model means removing old, unused features and adding new features during long-term operation of the robot. In order to preserve the ability to use the updated spher-

ical views for the visual navigation system (see Chapter 4), the feature points of each node that need to be moved to the STM and LTM stores should be located on the reference sphere, as if these features were seen from the same point when the node was first created. This ensures that the robot keeps the ability to use the reference views for heading estimation and can therefore navigate using the map.

In order to achieve this goal, the 3D position of feature points shared between one view from the current visit and one from the previous visit to the node are reconstructed. The current and previous views are each obtained by selecting an image in the recorded sequence for that visit with the highest similarity score to the reference view. The 3D position of the shared points can be determined to an unknown scale, as the norm of the translation vector is fixed to unity. These points are divided into three groups: those which already exist in the LTM store of the node, those which already exist in the STM store of the node and new points which need to be added to STM.

In order to add new features in STM into their correct position on the sphere, a simplified version of what is known in the computer vision literature as a "multi-baseline stereo" (Kang and Szeliski, 1997) is used. In our case, only two stereo pairs between three views are used: the reference view, the current view and a view from the previous visit to the node. These three views are captured in different visits to the node, the aim of which is to update the spherical view of the node by transferring feature points to it from the spherical views of the visits.

In Fig. 5.4, let X_o be one of the reconstructed positions for an image feature that is shared between the three views C_c, C_p and C_r, where C_r is the reference view of the node, C_c is the selected view from the current visit and C_p is the

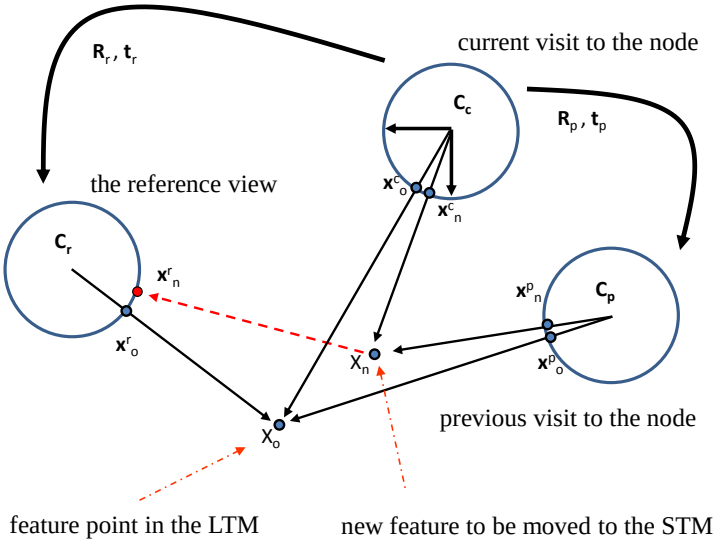

Figure 5.4: Reference view updating. Current view C_c is matched with previous view C_p and reference view C_r to estimate the coordinates of new features in the spherical representation of the reference view.

selected view from the previous visit.

Based on the stereo views (C_c, C_p), and for a 3D point X_o in the view of the robot, it is possible to write:

$$X_o^p = \lambda_p x_o^p, \tag{5.1}$$

where X_o^p is the representation of X_o in the reference frame of C_c, x_o^p is the projection of X_o^p onto the unit sphere of C_c and λ_p is the depth of X_o based on the unknown scale of the stereo views (C_c, C_p).

Also, in the reference frame of the view C_c and based on the stereo views (C_c, C_r), it is possible to write:

$$X_o^r = \lambda_r x_o^r, \tag{5.2}$$

where X_o^r is the representation of X_o in the reference frame of C_c and λ_r is the depth of X_o based on the unknown scale of the stereo pair (C_c, C_r).

Equations 5.1 and 5.2 show that 3D point X_o, shared between the three views, has two depth values, λ_r and λ_p, depending on the unknown scale of each 3D reconstruction. This also means that it is possible to convert between the two unknown scales as follows:

$$X_o^r = sX_o^p, \quad s \in \mathbb{R}. \tag{5.3}$$

The value of s is estimated such that it minimises the distance error between the 3D point's correspondences between the two stereo pairs (C_c, C_p) and (C_c, C_r).

Now, let X_n be the reconstructed position of an image feature shared between the two views C_c and C_p, although it does not exist in view C_r. In order to find the projection of X_n onto the sphere of C_r, first the depth of X_n needs to be converted to the scale of the stereo pair (C_c, C_r) using s:

$$X_n^r = sX_n^p, \tag{5.4}$$

where X_n^p is the representation of X_n in the reference frame of C_c based on the scale of the stereo views (C_c, C_p) and X_n^r is the representation of X_n in the reference frame of C_c based on the scale of the stereo views (C_c, C_r).

Next, as shown in Fig. 5.4, view C_c and reference view C_r are related by a rigid body displacement represented by the rotation matrix $\mathbf{R}_r \in SO(3)$ and translation $\mathbf{t}_r \in \mathbb{R}^3$ (see Chapter 3 for details on the estimation of $\mathbf{R}_r$ and $\mathbf{t}_r$). It

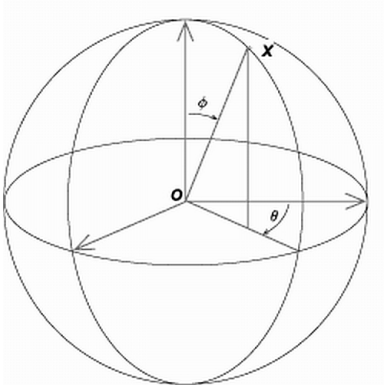

Figure 5.5: Spherical coordinates.

is possible to transform X_n^r to the frame of the reference view C_r as follows:

$$X_n^c = \mathbf{R}_r X_n^r + \mathbf{t}_r. \tag{5.5}$$

Finally, the point x_n of the new feature can be located on the unit sphere of the reference view by normalisation:

$$x_n = \frac{X_n^c}{\|X_n^c\|}. \tag{5.6}$$

5.4.3 Recursive Filtering

A key issue for any long-term mapping and localisation system is the danger that the interdependency between mapping and localisation will introduce a positive feedback loop, where measurement noise may be picked up and amplified to the point where the system becomes unstable (see related discussion in Ess et al. (2008)). An important question is therefore how to avoid such instabilities and maintain robust, long-term performance.

In the proposed hybrid mapping system, new features will not be recorded in

exactly the same location as the original features, so they have to be projected onto the spherical view representation, as described above. As the multi-view geometry calculations will be sensitive to noise, the process of adding new points could introduce an accumulated error, which would eventually result in the degradation of map quality. To mitigate these effects, it is possible to exploit the fact that the position of each new added feature can be re-estimated repeatedly during rehearsal and recall, meaning that a recursive filtering method can be applied to produce estimates that will tend to be closer to the true values of the measurements. So, if the position of the point is represented by its spherical coordinates (see Fig. 5.5), as

$$\mathbf{x} = [\theta, \phi]^T,$$

then it is possible to formulate the problem as an estimation problem with $\mathbf{x}$ as the state of the system.

The process model for such a system is simple where the state is unchanged as soon as it is added to the STM. The measurements for this state will come as 3D points on the sphere, and the observation model, which relates these measurements to the state vector, will involve the following non-linear mapping:

$$\mathbf{z} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} \sin(\theta)\sin(\phi) \\ \cos(\theta)\sin(\phi) \\ \cos(\phi) \end{bmatrix}, \tag{5.7}$$

where z is the observation vector. Due to this non-linearity, it is possible to consider this as a simple non-linear estimation problem, and therefore use any suitable technique such as the Unscented Kalman Filter (UKF) (Julier and Uhlmann,

1997) in which a small number of carefully chosen sample points are propagated in each estimation step. For a state space with dimension L, $2L + 1$ points are selected such that their sample mean is the state vector and their covariance is the process covariance (as such, five points were used in the experiments presented later). The non-linear function is applied to each point in turn to yield a cloud of transformed points that provide a compact parametrisation of the underlying distribution. This filter was chosen over other non-linear filters such as the Extended Kalman Filter (EKF) (Uhlmann, 1994) due to its more accurate estimation properties.

Section 5.6.1 presents an experiment conducted to test the long-term stability of the system when updating features on the spherical view representation, with and without the recursive filtering step. The results show the effect of accumulating measurement errors on the system and the reduction of this error when the UKF is used.

5.5 Calibration of Model Parameters

This section addresses the temporal calibration of the system, i.e. the unit of time that determines the stages of memory, following which it provides a discussion about the effect of choosing different numbers of stages for the STM and LTM stores.

5.5.1 Temporal Calibration

Temporal calibration means selecting the real-time unit in which the robot uses the memory system to update its map. In this work, it is assumed that the system

is used by a mobile service robot working inside a house or public environment, where life is a series of daily episodes. This suggests that using days as a basic time unit would be a realistic choice. After each working day, during which it spends its time navigating inside the environment and visiting different nodes inside the map, the robot goes through what could be called a "sleep" period where it activates its memory system to update the map. However, in other situations where life and human activities do not follow daily cycles, the robot can adopt a time scale to update the map which reflects the natural cycle of activities in its surroundings.

Depending on the nature of the task, the robot may visit some nodes in the map multiple times during one day, whereas other nodes will be visited less frequently. This means that it is important to unify the rate at which the appearance of each node is updated. In order to achieve this aim, the robot selects, at the end of each working day and for each visited node in the map, one view only to use for map updating. The selected view is the one which has the highest number of matched points with the reference view of each visited node over the whole day.

5.5.2 Number of Stages in STM

The number of stages in STM affects the rate at which image features move from STM to LTM, in that more stages in STM, the less features will be transferred to LTM. On the other hand, the fewer the number of stages in STM, the more features will be transferred to LTM. The target is that only stable and descriptive image features are selected for moving from STM to LTM. Stable features refer

to those features which can be observed easily and can be used as a signature to describe the appearance of the environment. For example, features coming from a picture on a wall are a good signature for a room, as the picture is elevated and rarely moved, whereas features coming from objects such as chairs and other movable furnitures, which we use daily inside our living spaces, are not stable and should not be used as a signatures for describing the appearance of a place, as they are subject to movement and can be obscured easily.

STM is designed to select stable and descriptive image features by allowing only features which are persistent and successful in appearing repeatedly to advance through the stages of STM and reach LTM. However, it is also important not to have a large number of stages in STM; otherwise, the map will fail to adapt quickly to new changes in the appearance of the environment. Therefore, and in order to analyse the behaviour of the memory model when using different numbers of stages in the STM and establish a suitable number of stages for STM, an experiment is presented in Section 5.6.2.2, where the transfer rate between STM and LTM is studied while the same experiment is repeated with different number of stages in STM.

5.5.3 Number of Stages in LTM

The number of stages in LTM affects the total number of features used to represent each node in the map. If the number of stages is too great, the map will have a long memory and the features will stay in the memory store, even if they have disappeared from the view of the environment. The risk in this case comes from the fact that these features are now considered as noise and can reduce the

localisation and navigation accuracy of the robot using the map.

On the other hand, if the number of stages is too short, the map will tend to forget useful information too quickly, and the number of features which represent each node in the map will become very small. The risk in this case comes from the fact that with few features to represent the node, the localisation accuracy can also be affected.

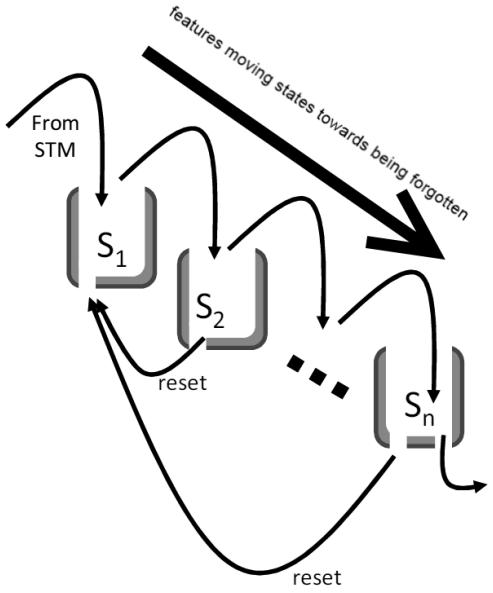

Figure 5.6: The flow of image features in LTM.

As mentioned earlier, the updating process of the map is carried out according to a fixed time step, which changes the state of each feature point inside the memory stores. Some features will move the stage towards being forgotten if they are absent, and others will be reset to the first stage as soon as they are matched with features from the new view of the node. As soon as any feature

in LTM is matched with one of those from the robot's current view, the feature is reset to the first stage, which tends to happen in situations where part of the scene is blocked for some time and then re-appears again. This means that from each stage in LTM, apart from the first stage, different numbers of features are reset to the first stage after each time step, i.e. a reset feed (see Fig.5.6). As the features move away from the first stage, their chance of being reset to the first stage reduces, lessening the overall number of features coming back from the stages, which eventually goes to zero, as the features move away from the first stage. Therefore, the reset feed of features to the first stage can be used as an indicator for choosing the number of stages in the LTM store. This can be done by choosing the number of stages that provide a reset feed above a certain level. In Section 5.6.2.1, an experiment with different numbers of stages in LTM is presented, where the reset feeds that return from each stage are analysed, showing decreasing reset feeds as the features move away from the first stage of LTM.

5.6 Experimental Evaluation

5.6.1 Long-Term Stability of the System

In order to test the long-term stability of the system when updating features on the spherical views of the map, the following experiment was designed. The robot first captured a reference view in the middle of a room, and then it was rotated approximately 90^o with respect to the reference view. After that the robot was moved back and forth, covering approximately 1 m on each side of the location

Figure 5.7: Two views of the room where the experiment took place. The appearance of the room was changed manually during the experiment by moving objects and office dividers around the robot.

where the reference view was recorded. While the robot was moving, a set of 251 images was recorded to capture the changing appearance of the room, which was changed manually over the course of the experiment.

In order to test the accuracy of estimating the relative heading between the 251 images and the reference view in the middle of the room while using the memory model to update the spherical view representation, a copy of the 251 images was made by reversing the sequence of the images, and then the reversed sequence was appended to the original dataset. The result of this operation is a single dataset representing one run of the robot forwards and backwards through the environment (see the top row in Fig. 5.8). The appending of the two sequences means that measurement errors in one direction should be balanced by equal and opposite errors in the opposite direction, so that net drift in the estimated orientation of the robot after one such run should be zero (thus providing the reference standard for this experiment, i.e. an accumulated error of zero would represent a "perfect" result after a large number of repetitions of the same run). The image sequence was looped multiple times and used to test the updating mechanism for the reference view as a single map node (see the second and third

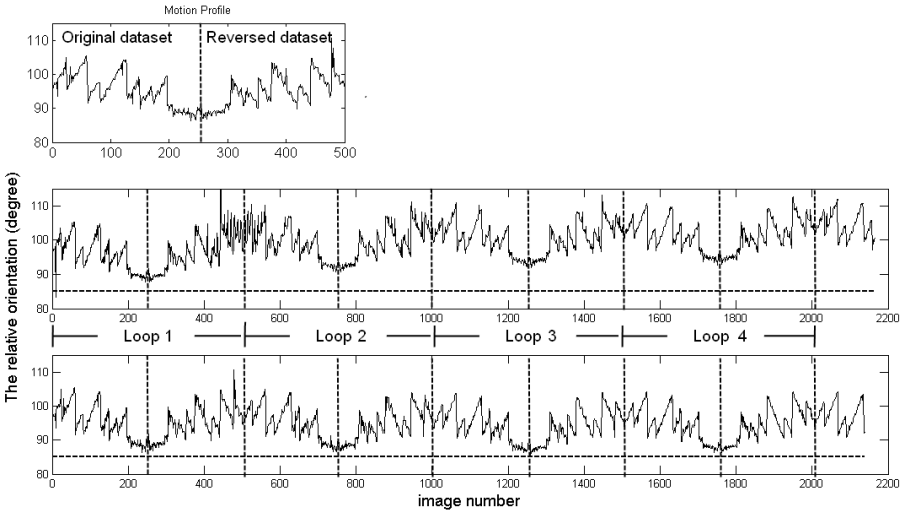

Figure 5.8: The top row shows the estimated heading relative to the reference view, using the original dataset combined with the reversed dataset. The second row shows how the estimation of the heading without the UKF drifts over time due to noisy measurements, where the combined dataset was looped repeatedly. The last row shows the effect of the UKF where long-term drift is greatly reduced.

rows in Fig. 5.8). By considering each image in the looped sequence as a daily visit to the node, the total number of visits was around 2,200, which is approximately equivalent to a period of 72 months.

The stability of the proposed updating mechanism can be judged based on the relative heading between the recorded images and the reference view in LTM. The second row of Fig. 5.8 shows the heading sequence estimated from around 2,200 images, but without applying the UKF filtering step from Section 5.4.3. The effect of the accumulated error is obvious here, making the motion profile drift gradually over successive loops of the dataset. The third row of Fig. 5.8 shows the effect of the UKF step, where the drift is noticeably reduced.

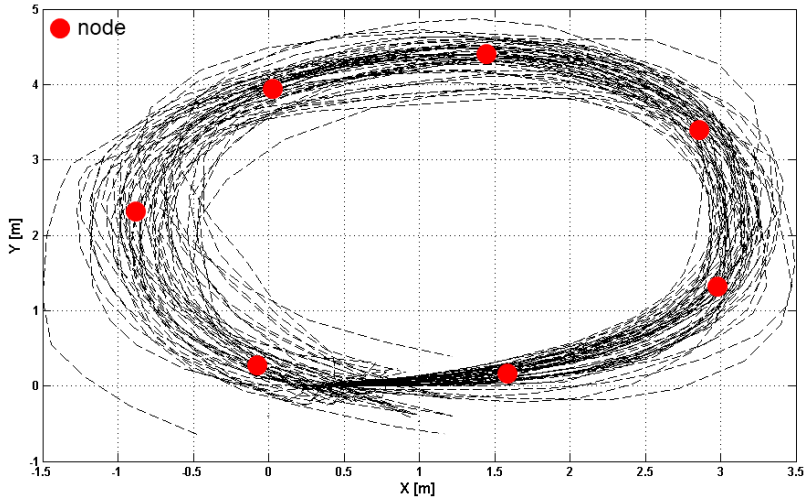

Figure 5.9: Trajectory of the robot during 50 tours inside a robotic lab at the University of Lincoln obtained from laser-corrected odometry. The first tour was used to create a map consisting of seven nodes (selected manually). The dataset of images generated from these tours is used for both the experiment presented in Section 5.6.2 and the experiment presented in Section 6.2.

5.6.2 Different Numbers of Stages in LTM and STM

This experiment was conducted to study the effects of using different numbers of stages in the LTM and STM stores. The focus of the experiment is the reset feed of image features in LTM (as explained previously is Section 5.5.3), as well as the rate at which the features move from STM to LTM.

A set of 1385 images was captured by the camera on board the robot while driving it inside a robotics lab at the University of Lincoln (see Fig. 6.1). The images were generated from 50 tours of the lab, where the robot was driven by hand (see Fig. 5.9). After each tour the appearance of the lab was changed manually to simulate the activity of a working day. The changes involved the arrangement of objects inside the room, including adding new objects like boxes and posters, removing existing objects individually and also covering them with

movable office dividers for certain periods. The first tour was used to create the map consisting of seven nodes (selected manually). The rest of the image sequence was used as an input for the localisation system. Global localisation based on place recognition and using the similarity score between the current and the reference views (winner-takes-all) was used in this experiment. After each tour the nodes of the map were updated using the memory model. To study the effects of using different numbers of stages on the reset feeds of LTM and the transfer rate from STM to LTM, the memory model was tested with different numbers of stages in LTM, while keeping a fixed number of stages in STM. Subsequently, the opposite was done, whereby memory was tested with different numbers of stages in STM, while the number of stages in LTM was fixed.

5.6.2.1 The Reset Rate inside the LTM

In order to study the influence of the reset feeds in LTM, the images generated from the 50 tours were used. After each tour, and for each LTM store in each node, the rate at which each stage in LTM feeds back to the first stage was recorded. This rate was computed as the percentage of the total number of features which fed back to the first stage. At the end of the 50 tours, the total reset rate for each stage was computed as the median for all the reset rates recorded during all 50 tours.

Let $f_{s,n}^t$ be the number of features returned from the stage number s to the first stage of the LTM store of node number n during tour number t, and let F_n^t be the total number of features fed to the first stage from all the stages in the

LTM store of node number n during tour number t,

$$F_n^t = \sum_{s=2}^{m} f_{s,n}^t, \tag{5.8}$$

where m is the number of stages in LTM.

Then the reset rate for stage number s of the LTM store of node number n for tour number t can be computed as follows:

$$r_{s,n}^t = \frac{f_{s,n}^t}{F_n^t} * 100, \tag{5.9}$$

while the reset rate for stage number s for all the tours is computed as follows:

$$R_{s,n} = median(r_{s,n}^t),\ t \in 1...T, \tag{5.10}$$

where T is the total number of tours.

In order to test the effect of choosing different numbers of stages in LTM, the procedure for updating the map using the image sequence from the 50 tours was repeated 7 times with different numbers of stages in LTM, while the number of stages in STM was kept fixed.

For each repetition the number of stages in LTM was as follows: 4, 5, 8, 10, 12, 14 and 20. Fig. 5.10 shows an example taken from the third repetition of the experiment, where eight stages were used in LTM and three stages in STM. The figure shows $R_{2,4}$, $R_{3,4}$, $R_{4,4}$, $R_{5,4}$, $R_{6,4}$, $R_{7,4}$ and $R_{8,4}$ for the eight stages of the LTM store of node number 4 in the map.

Fig. 5.11 illustrates the reset rates for all seven nodes in the map for the case when eight stages were used in LTM and three stages in STM. Although the

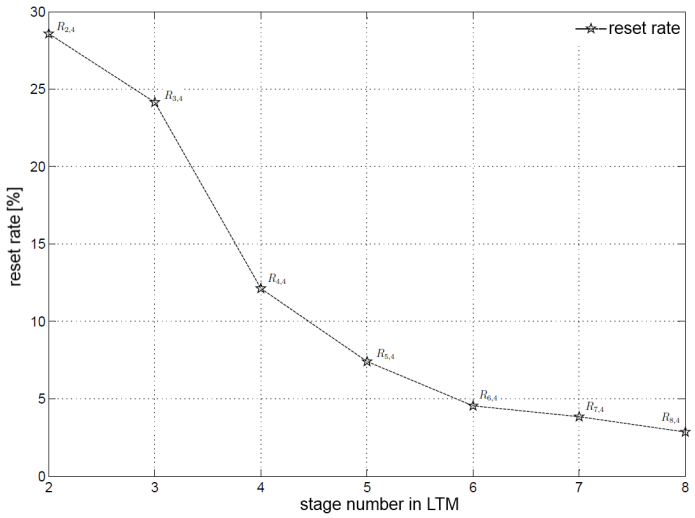

Figure 5.10: The reset rate from each stage in the LTM store of node number 4 in the map when eight stages in LTM and three stages in STM were used.

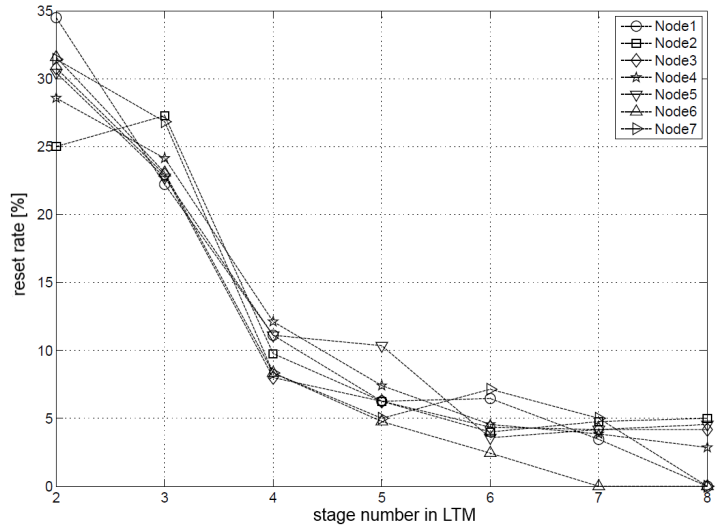

Figure 5.11: The reset rates for all the nodes in the map when eight stages in LTM and three stages in STM were used.

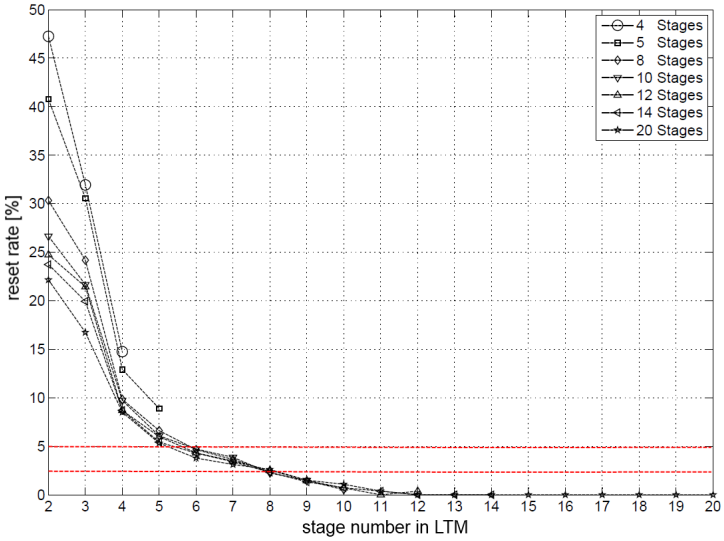

Figure 5.12: Reset rates for all the seven repetitions of the experiment, i.e. 4, 5, 8, 10, 12, 14 and 20 stages in LTM, while using three stages in STM.

appearance of each node in the map was subject to different changes, the reset rate from the stages in LTM shows similar characteristics where it starts around 30% from the second stage and drops to less than 5% in the last stage.

As mentioned above, the experiment was repeated with different numbers of stages in LTM. For clarity, and in order to visualise the data, the reset rate from each stage in LTM is computed as the average for all reset rates from that stage in all the nodes in the map:

$$R_s = \frac{1}{N} \sum_{n=1}^{N} R_{s,n} \tag{5.11}$$

where N is the total number of nodes in the map.

In Fig. 5.12, R_s is plotted for all the seven repetitions of the experiment, i.e. 4, 5, 8, 10, 12, 14 and 20 stages in LTM, while using three stages in STM. The figure shows that the reset rate decreases as the numbers of stages in LTM

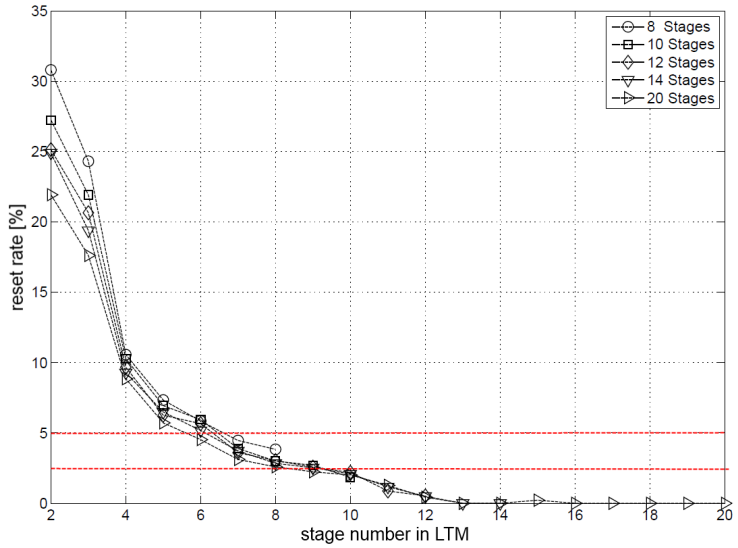

Figure 5.13: Reset rates for five repetitions of the experiment, i.e. 8, 10, 12, 14 and 20 stages in LTM, while using two stages in STM.

increase. The rate drops to around zero after stage 9. Fig. 5.13 shows the same results when two stages were used for STM instead of three.

Based of the above results, the choice of six, seven or eight stages for LTM would be reasonable if the reset rate from the last stage was chosen between 2.5% and 5%. These values give the best results in the sense that the memory is not too short that useful features would be forgotten very quickly and also not too long that old features that have disappeared from the scene would be kept in the memory.

5.6.2.2 Transfer Rate from STM to LTM

In order to choose the number of stages for STM, the transfer rate of image features from STM to LTM was analysed. Different numbers of stages in STM with a fixed number of stages in LTM were used for the same experiment from

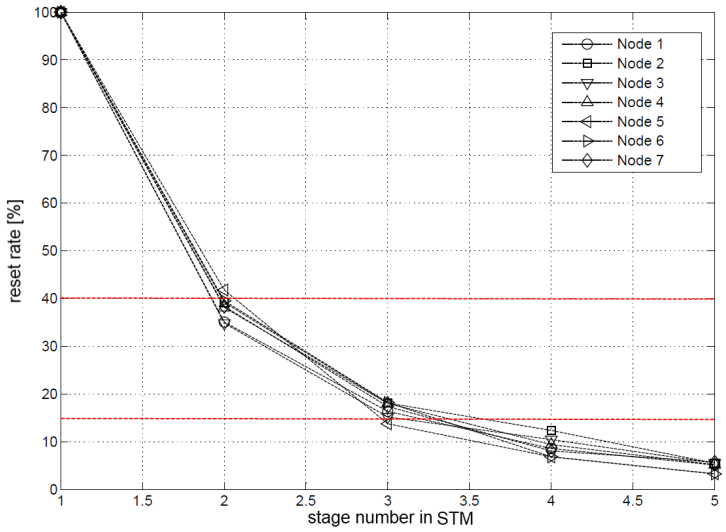

Figure 5.14: Transfer rates for different number of stages in STM.

the previous section. The LTM was set for this experiment at 7 stages and the experiment was repeated with 1, 2, 3, 4 and 5 stages in STM.

Similar to the analysis of the reset rate in LTM in the previous section, after each tour and for each node in the map the rate at which STM feeds into LTM was recorded. This rate was computed as a percentage of the number of features which move from STM to LTM to the total number of features which enter the last stage in STM. The total transfer rate for each stage is computed as the median for all rates recorded during the 50 tours.

Fig. 5.14 shows the transfer rate for all STM stores in the 7 nodes in the map. When the system was tested with only one stage in STM, all features entering STM were moved to LTM and the rate was 100%. As the number of stages in STM increased, the rate of the image features moving to LTM decreased and fewer features were moved to LTM. When 5 stages were used for STM, very few features that entered STM managed to find their way to LTM.

Deciding to choose the number of stages in STM, where the transfer rate from STM to LTM is between 40% and 15%, leads to the reasonable choice of two or three stages.

Chapter 6

Experimental Evaluation

This chapter describes experiments conducted to test the long-term performance of the system. The experiments evaluate the adaptability of the map and its consistency through performing self-localisation and visual navigation tasks inside changing environments over a period of up to nine weeks.

6.1 Introduction

In order to evaluate the proposed updating mechanism, the following experiments were conducted to assess the adaptability of the map and its consistency over time.

- **Reference Views Adaptability**: The main goal of the proposed memory model is to adapt the reference views of the map according to changes in the appearance of the environment over time. In order to measure this adaptability, the similarity between the views recorded by the robot when visiting the nodes and the reference views of these nodes in the map was

used as a metric. Higher similarity indicates a more accurate representation of the environment. The change in the similarity was compared over time when the memory was used and when the reference views stayed unchanged.

- **Map Consistency**: Consistency in our case means that the updating process, which involves adding and removing image features to and from reference views over time, does not cause the map to degrade, i.e. accumulated errors during the process of updating the spherical views lead to misplacing newly added image features on the surface of the reference spheres. If the map degraded over time, the robot would have difficulties using it for tasks such as autonomous visual navigation. Therefore, measuring the performance of the robot in executing a visual navigation task over a long period of time would be a good indicator of the quality of the map, which is considered consistent if navigation performance does not degrade over time.

The following sections in this chapter provide the results of four experiments conducted in three different environments – a robotic lab, a restaurant hall and the floor of an office building.

1. The first experiment took place inside a laboratory room. The experiment was designed to test the adaptability of the map when faced with large changes in the appearance of its surroundings. The changes were made manually during the course of the experiment.

2. The second experiment was designed to test the localisation accuracy of the map when using the updating mechanism in the presence of noise, which was generated automatically, and the system was tested using a Monte

Carlo simulation technique. The experiment took place in a restaurant hall over a period of 9 weeks.

3. The third experiment was designed to test the overall performance of the system under real operating conditions, and was conducted inside a large and real changing environment over a period of approximately two months. The environment was an office floor, where different staff activities affected and changed the appearance of the environment. The experiment provides a comparison of localisation accuracy between a map with static reference views and a map with the proposed adaptive view representation. Also, the experiment measures errors in estimating the heading of the robot relative to the reference views in the map.

4. The fourth experiment was designed to test the consistency of the map over time. The performance of the robot was measured while executing a daily visual navigation task using the reference views of the map. The results showed that the map does not degrade as a result of the continuous updating of its reference views. The experiment took place in a faculty administration office over a period of 8 weeks.

During all the experiments, global localisation based on place recognition was used by employing a similarity score between the current view of the robot and the reference views in the map (winner-takes-all), in order to locate the robot in one of the nodes in the map.

In order to obtain the ground truth positions for the images recorded by the robot during the experiments, each image was recorded along with a laser scan and odometry reading. This data enabled us to use a standard open source laser-

Figure 6.1: Two panoramic views from approximately the same place in the laboratory environment but at different times.

based SLAM algorithm, the GMapping library (Grisetti et al., 2007a), to correct the odometry reading of the robot and attach each recorded image with a 2D position and a rotation relative to the robot's starting point. The output of the algorithm was an estimate of the robot's trajectory, along with an occupancy grid map of the environment. It is important to emphasise that the method presented in this work did not build or require an occupancy grid map of the environment. This information was used only for ground-truth validation and visualisation.

6.2 Manually Modified Environments

Using the same dataset as in the experiment presented in Section 5.6.2, this experiment was carried out in a robotics lab at the University of Lincoln, where a set of images was collected by driving the robot in a loop. The images were generated from 50 tours over a period of three days, resulting in 1385 images. After each tour the appearance of the lab was changed manually. The changes involved the arrangement of objects inside the room, including adding new objects

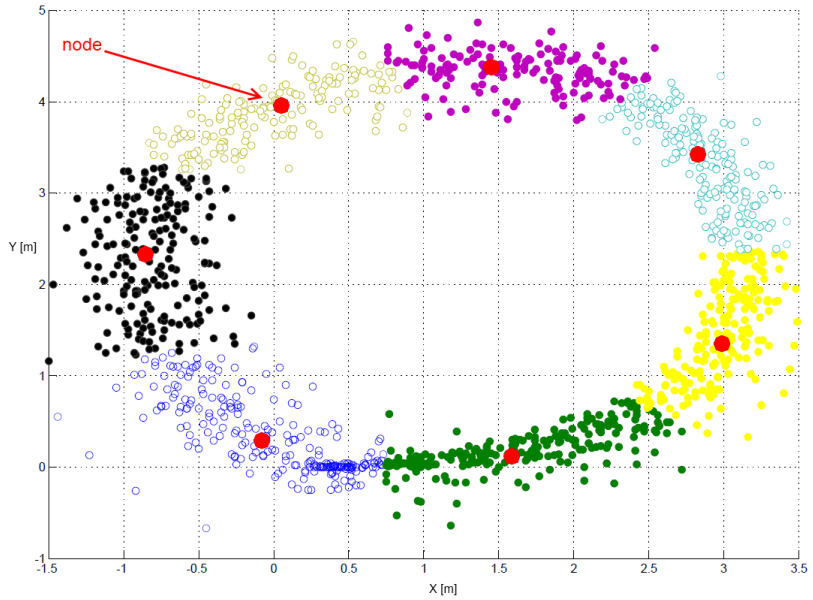

Figure 6.2: Ground truth positions of the recorded images obtained from laser-corrected odometry. The constructed map consists of 7 nodes, each colour representing the group of images which belong to the same node.

like boxes and posters, removing existing objects individually and also covering them with movable office dividers for certain periods. Fig. 6.1 shows two images taken from approximately the same place at different times during the experiment. In order to obtain the ground truth positions for the collected data, the starting points for all 50 tours were initialised from a fixed and known place inside the room, while each image was recorded along with laser scans and odometry.

The first tour was used to create a map consisting of seven nodes, which for this experiment were selected manually with around 1.5 m between consecutive nodes. Fig. 6.2 shows the ground truth positions of all the collected images, each colour representing a group of images belonging to the same node. The rest of the image sequence was used as an input into the localisation system.

During a visit to a node in the map, the robot will capture a number of images

as it goes through the node. Among these images, the image with the highest similarity score is used to represent the view of the node for that visit, and it is then used to update the reference view using the proposed updating mechanism. Fig. 6.3 shows how the similarity score changed over the 49 tours for a selected node in the map. The similarity score is the percentage of the matched feature points between the current view of the robot and the reference view of the node. As shown, using the adaptive views gave a higher similarity score, while for the static view the similarity score sometimes dropped below 10%. Fig. 6.4 shows the mean of the similarity score for each node in the map over all the tours. The results show an increase of approximately 15% in similarity scores when the adaptive views were used in the map.

After the localisation step, the reference view of the node and the input image were used to estimate the rotation between the two views using the method from Section 3.3.2.2, and then the estimated rotation was compared with the ground truth data obtained from laser-corrected odometry. Using the sequence of collected images, the same experiment was repeated once using static reference views for the map and then using the multi-store memory model, both with and without the Unscented Kalman Filter, which is described in Section 5.4.3. Static reference views were created from the first run (similarly, the first run was used to initialise LTM) and subsequent runs were used for localisation.

Table 6.1 shows a comparison using several performance measures, exhibiting mean and standard deviation, between the static and the adaptive approach using eight stages for the LTM and three stages for the STM. As shown in the first row, errors in the estimation of the rotation did not drop significantly, whereas the average number of matched points for the winning node, which is used for global

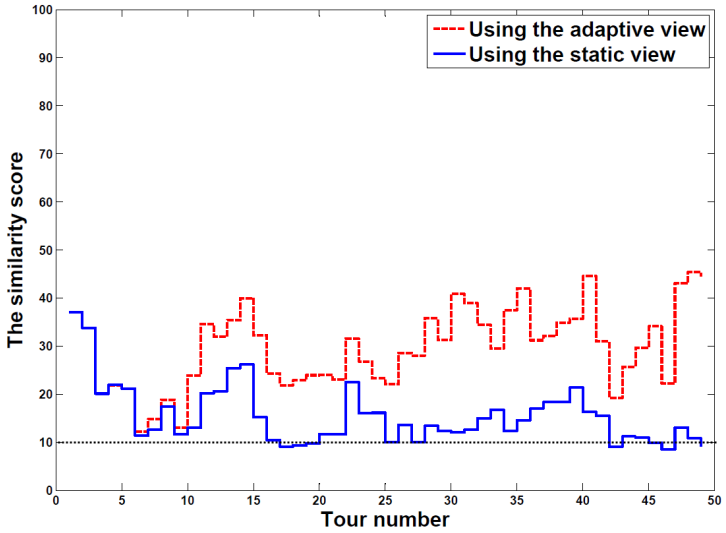

Figure 6.3: The similarity score for node 4 using the static view and the adaptive view during all the tours.

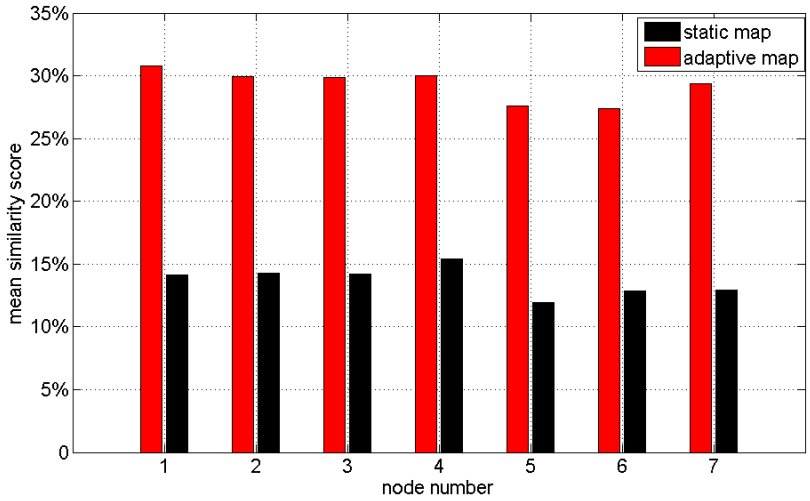

Figure 6.4: The mean similarity score for each node in the map over all the tours.

localisation, increased noticeably when the memory model was used, as shown in the second row. As the environment changed over time, the number of matched points for the winning node became smaller when the static reference views were used. As shown in the third row, during global localisation the winning number of matched points was over 50 in 77.0% of cases for the static map and 95.1% for the adaptive memory model.

Table 6.1: Comparison Measures

Metrics	Static	Memory -UKF	Memory +UKF
Error in estimating rotation	$4.2^o \pm 4.1^o$	$4.5^o \pm 4.6^o$	$4.0^o \pm 4.1^o$
Mean number of matched points	81 ± 43	118 ± 54	120 ± 55
Matched points > 50	77.0%	95.1%	95.1%

6.3 Localisation Performance with Added Noise

For this experiment, a simple map of the students' restaurant in the University of Lincoln was created by taking eight omni-directional images from eight different places to form the reference views for the nodes. This restaurant is used for various student activities, and between these events the place is generally returned to its normal appearance by the restaurant staff but with some differences.

Over a period of approximately nine weeks the robot visited the eight locations 18 times and recorded images for places in the map. Fig. 6.5 shows two panoramic images for the same place recorded at different times. Using 144 images generated from these visits, the method for adapting the reference views inside the map was tested by using a Monte Carlo simulation method (Rubinstein and Kroese, 2008). Localisation failures were an integral part of this experiment, i.e. in the case of

Figure 6.5: Two panoramic views from the same place in a university restaurant at different times.

incorrect place recognition the image representation for the wrong node would be updated, and Monte Carlo simulation was used to simulate missing information and added noise due to factors such as illumination changes in the current view. One hundred trials were used for every test to evaluate localisation performance, while five different tests were carried out using the restaurant dataset with four stages in the LTM and two in the STM. The results from these tests are presented in Table 6.2. The length of LTM store used for this experiment is shorter than the LTM used for the other experiments presented in this work. This is due to the different nature and dynamic of the open-space restaurant hall environment where the experiment took place compared to the other indoor office environments.

In the first test, the localisation accuracy for 144 images was measured without any simulated missing information or added noise. In the second test, a simulated missing information was produced by removing 50% of randomly chosen extracted features from the current view before each of the 144 localisation attempts. In the third test, localisation performance was measured with noise in the matching

Table 6.2: Localisation Accuracy with Added Noise

	Correct global Localisation [%]	
	Static Map	Adaptive Map
no noise or missing information	95.8	98.6
50% missing information, no noise	93.4 ± 1.2	98.4 ± 0.8
added noise, no missing information	89.7 ± 1.8	97.2 ± 0.9
25% missing information + noise	88.0 ± 1.8	96.2 ± 1.7
50% missing information + noise	85.6 ± 2.1	93.7 ± 2.4

scheme by adding Gaussian noise $\mathcal{N}(0, 0.1)$ to the distance threshold between the nearest and second nearest neighbour of the image feature's descriptor. By adding this noise some of the true matches would be missed and some of the false matches counted. In the fourth test, the two factors were combined: 25% of randomly chosen extracted features were removed from the current view before the localisation stage, and then Gaussian noise was also added into the matching scheme. In the last test, 50% of randomly chosen extracted features were removed, and then the Gaussian noise was added.

The observed performance differences between static and adaptive mapping were tested using Student's t-test, and were shown to be statistically significant ($p < 0.01$). The results show that the adaptive map yields better localisation performance due to its more accurate representation of the real appearance of the environment.

6.4 Large Office Floor Environment

This experiment was carried out in the real changing environment of an office floor at the University of Lincoln. This area is used for student and staff ac-

tivities. Over a period of approximately two months, the robot was driven on tours between the offices while recording a set of images. The result was 36 tours (one tour per day) generating 6161 images, with approximately 35cm between consecutive images. The first tour was used to create a map of 21 nodes.

Fig. 6.6 shows the ground truth positions of the recorded images. Each colour represents the group of images which were assigned to the same node. The grey circles represent the reference view for each node. The first tour was used to select nodes of the map using the dual clustering method from Section 4.3.1. The similarity threshold used for dual clustering was 35 image features and the distance threshold was 1.5 m. The image similarity threshold Ψ_s was assigned to 35 feature points for this experiment because as the results showed that when fewer than 35 feature points are used to estimate the essential matrix between two spherical views, a degenerate solution for the essential matrix or a very noisy estimation for the heading could be obtained.

For each subsequent tour, the robot's true trajectory through the environment was estimated, and then the output map was aligned with the first tour map in order to transform ground truth navigation trajectories into a common reference frame.

The method used for adapting reference spheres in the map, including the Unscented Kalman Filter, was tested using the collected images. The tests were carried out using the memory model with eight stages in the LTM and two in the STM. The results show that the localisation failure rate stood at around 1.5% when the adaptive approach was used, whereas the failure rate for its static counterpart was 3.5%. More importantly, the experiment demonstrates the ability to use image features which match between the reference view and current image

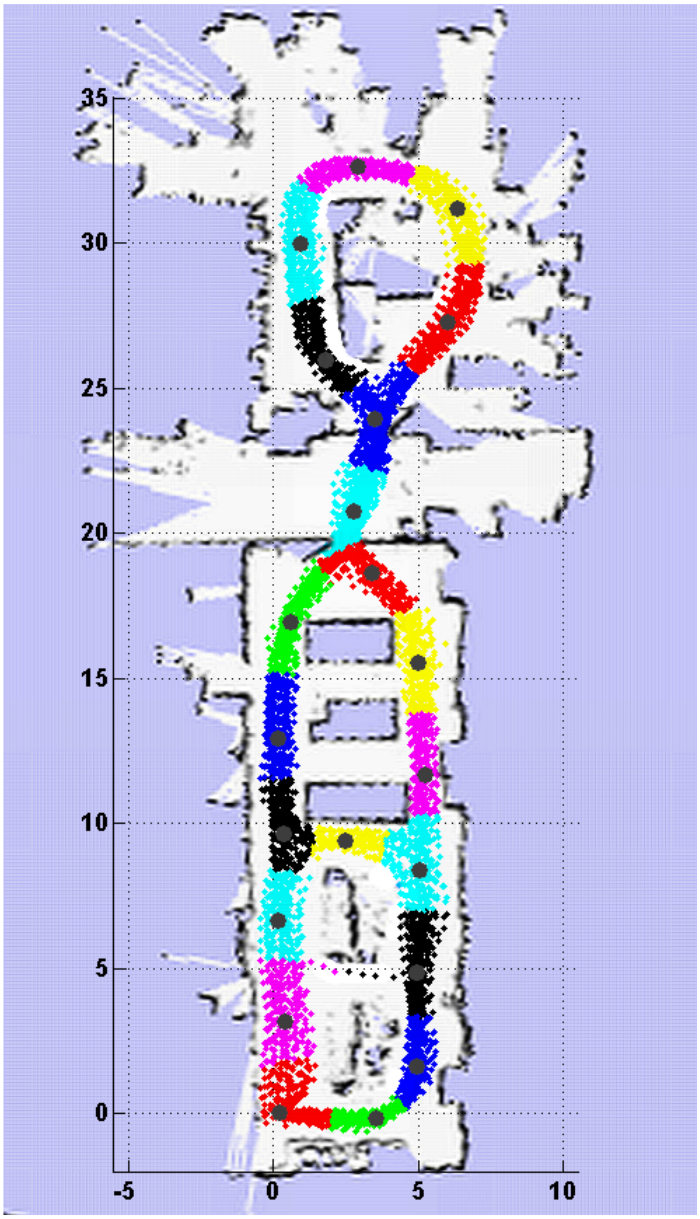

Figure 6.6: Ground truth positions of the recorded images obtained from laser-corrected odometry in the office environment. The constructed map consists of 21 nodes, where each colour represents the group of images which were assigned to the same node. The grey circles represent the position of the reference view for each node.

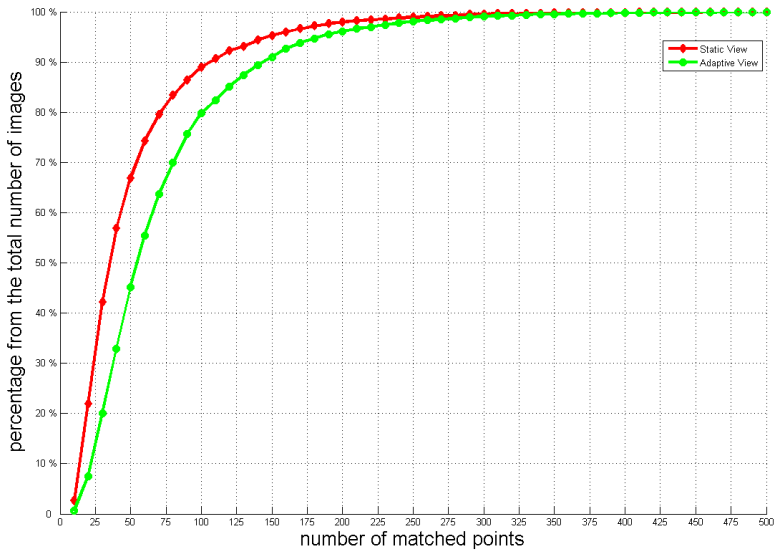

Figure 6.7: The cumulative percentage of matched points used for global localisation. The red line illustrates the results when static reference views were used for the map, whereas the green line corresponds to the adaptive views.

of the robot, in order to estimate the appropriate heading. With static views, rotation could not be generated around 15% of the time, whereas the failure rate was only 5% for the adaptive views. This happened because the robot needs a sufficient number of matched features (more than 35); otherwise, a degenerate solution for the essential matrix or a very noisy estimation for the heading could be obtained. As the results show, and due to the changing nature of the environment, along with the missing information effect, the multi-store memory model provides better representation for the appearance of the nodes, thus leading to improved performance in localisation.

Fig. 6.7 shows the cumulative percentage of matched features used for localisation. The red line illustrates the results when static reference views were used for the map, whereas the green line corresponds to the memory model. The adap-

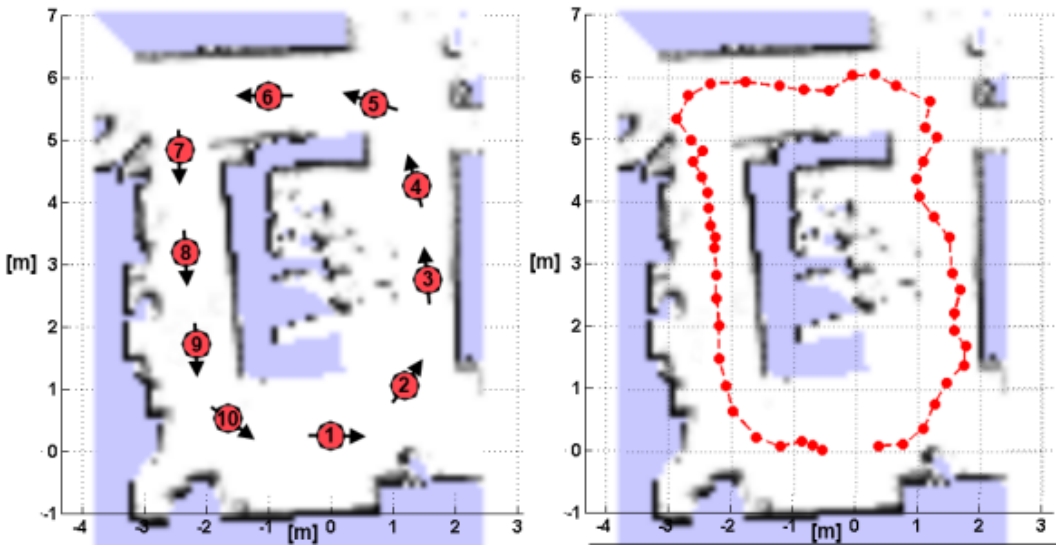

Figure 6.8: Left: a laser-based occupancy map for the area of the room where the experiment took place, showing the position of the nodes. Right: the trajectory of the path taken by the robot at day 34 of the experiment.

tive approach provides better matching scores, leading to better localisation and navigation performance. For example, 43% of the time the number of matched points between the reference view and the current image from the robot was below 35 points for the static views, whereas this happened only 24% of the time with the multi-store memory model.

6.5 Map Consistency

In order to evaluate the consistency of the map, the performance of the robot was measured while repeating a visual navigation task over a period of eight weeks. The robot followed the visual navigation strategy presented in Chapter 4.

Three metrics were used to evaluate the map. The first was the similarity between the reference views stored in the map and the views which the robot

used for navigation. The second metric was the length of the trajectory, i.e. if this increased over time, it would mean that the robot took more steps to execute the path due to poor directional information from the map. The third metric was the curvature of the executed trajectory by the robot, where the lower the curvature, the smoother the trajectory. The smoothness of the trajectory is a good indicator of the consistency of the decision-action relationship in a navigation system. Similar to the second metric, if the curvature of the trajectory increased over time, this would indicate that the robot performance was degrading.

The trajectory of the robot although is not a function between x and y, it is a curve in 2D plane and therefore it is possible to represent this curve using a function as follows.

$$y = f(x), \tag{6.1}$$

the length of this trajectory can be calculated as:

$$L = \sum_{i=1}^{n} \sqrt{(x_{i+1} - x_i)^2 + (f(x_{i+1}) - f(x_i))^2}, \tag{6.2}$$

where $(x_i, f(x_i))$, i=1...n, are the points of the trajectory in the Cartesian plane. The curvature of the trajectory at any point can be calculated as

$$k(x_i) = \frac{f''(x_i)}{[1 + (f'(x_i))^2]^{\frac{3}{2}}}. \tag{6.3}$$

Using the above curvature factor, the smoothness of the trajectory can be measured as follows:

$$B_E = \frac{1}{n} \sum_{i=1}^{n} k(x_i)^2, \tag{6.4}$$

Figure 6.9: Three images recorded from the same place at different times. The appearance changes through the existence of new objects in the arena and the disappearance of others.

where B_E is called the "bending energy" (Selekwa et al., 2004), the energy needed to bend a rod to a desired shape. In essence, the lower the energy, the smoother the trajectory.

This experiment was carried out inside the faculty office at the School of Computer Science, University of Lincoln. This area was chosen to conduct a long-term experiment because the room is designed in an open-office format in which seven members of staff perform their daily activities. These activities result in changes to the appearance of the room over time. Fig. 6.8 shows a laser-based occupancy map generated by the GMapping library for the area of the room where the experiments took place.

On the first day, the robot was driven in a loop, and then ten nodes were

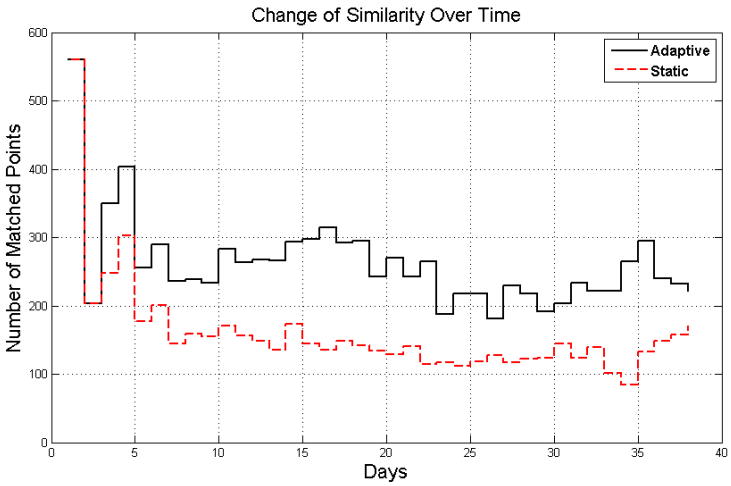

Figure 6.10: A comparison between the static and adaptive map, showing the change of similarity over the 38 runs for node number 4.

selected using the dual clustering algorithm from Chapter 4 to represent the appearance of the environment. Using these nodes, the map was used by the robot to perform a daily visual navigation routine from node number 1 to node 10. The ten-node route was repeated 38 times over a period of eight weeks, and after each run the robot used the memory model to update the map. The number of stages used in LTM and STM were eight and two, respectively. Fig. 6.9 shows three images taken by the robot from the same place but at different times.

At the beginning of each run, the robot was placed in the vicinity of the first node, after which it performed self-localisation by measuring the similarity between its current view and all the reference views in the map and localising itself to the node with the highest similarity score. Then the robot estimated its heading as described in Section 4.4.

As explained earlier in Section 4.5.2, simple obstacle avoidance tactic using sonar sensors was used so that the robot turned away from an obstacle by moving

a short distance toward a free space near that obstacle. If no free space was found, the robot moved backwards for a short distance.

As a result, the robot was able to perform the path-following task successfully during all runs. As mention earlier, the similarity metric was used as an indicator for the adaptability of the map. The mean number of matched features between the view with the best number of matching features and the reference views of the map was 170 ± 84 when static reference views were used for the map and 255 ± 92 when the adaptive map was used. This result shows the effect of using a memory model in increasing the similarity of the map to the environment. Fig. 6.10 shows how the similarity score changed over the 38 runs for node number 4 in the map.

The second metric is the change of trajectory length over time. Fig. 6.11 shows that it did not increase over time, staying in the range $19.9 \pm 0.8\ m$. In two cases, days 27 and 28, the robot took a longer trajectory due to a blockage in its usual route.

The third metric used for the evaluation is the smoothness of the trajectory measured by the bending energy. Fig. 6.12 shows that the bending energy of the trajectory did not increase over time and remained consistent. This implies that the quality of the map is also consistent over time.

6.6 Summary

This chapter presented the results from four experiments in which the proposed method for updating the reference views of a hybrid map was tested. The experiments were conducted in three different environments. The results of these experiments showed that the updating mechanism improved the similarity between

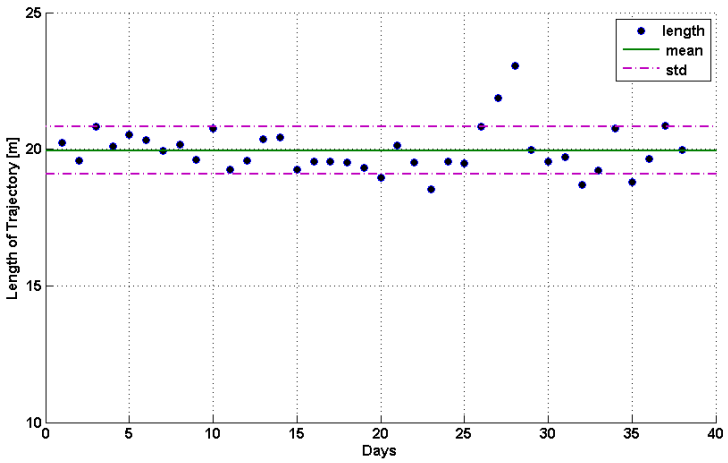

Figure 6.11: The change of trajectory length over time. The mean distance travelled over all runs was 19.9 ± 0.8 m. Between days 27 and 28 a big box was delivered into the office, taking up part of the robot's path and forcing it to take a longer trajectory.

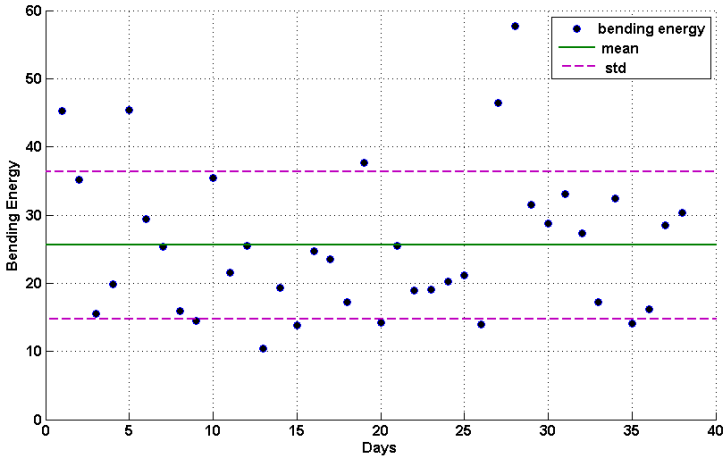

Figure 6.12: Change of trajectory smoothness measured by its bending energy.

the reference views of the map and the changing appearance of the environment over time. The results also showed that the map did not degrade over time and the robot was able to use the map to repeat a visual navigation task over a period of eight weeks.

Chapter 7

Conclusions and Future Work

This chapter presents a summary of the contributions of this thesis and a discussion about the limitations of the system and possible directions for future research.

7.1 Summary of Contributions

The work presented in this thesis addressed the problem of the long-term adaptation of a visual map by a mobile robot working in non-stationary environments. The robot adapted its map in response to changes in the appearance of its environment, and used the map for tasks such as self-localisation and autonomous navigation.

This work introduced a hybrid map representation for the robot's environment and was set on two levels, namely global and local. On the global level the world was represented as a sparse topological map, i.e. an adjacency graph where the nodes of the graph correspond to certain places in the environment. The aim is to represent the environment using a compact representation that targets mobile robots with limited computing resources. On the local level, a spherical

representation of the nodes in the topological map was introduced. Each spherical view contains a group of local image features mapped onto the surface of a unit sphere. This spherical representation is used as a connection between the global and local levels of the map. The group of image features contained inside each view is used as a descriptor for the appearance of the nodes, and is used for self-localisation on the global level of the map based on the similarity in appearance between the current view of the robot and the spherical views in the map. Similarity is measured according to the number of matched features between two views. After the localisation step, the robot uses the 3D bearing of the image features to estimate its current heading during visual navigation tasks. The heading is estimated using the multi-view geometry of spherical cameras.

The thesis also introduced an automatic method for selecting the nodes on the global level of the map. The spatial distribution of the nodes is determined using a dual clustering algorithm that uses two properties to select the nodes of the map – the geometric distance between the nodes and the image similarity between the spherical views of the nodes. The aim is to select a minimal set of nodes in particular locations, which provides the robot with complete coverage of the environment and prevents the occurrence of "gaps" in the map that could lead to navigation failures when using the map for navigation.

As a result of various human activities inside an indoor environment in which a mobile robot is required to operate, the appearance of the robot's surroundings is destined to change, leading to a change in the image features observed in its views of the environment. Over time, features which have been stored in the nodes when the map was first created will no longer represent the true appearance of the robot's world. To overcome this problem, an updating mechanism, which

adds and removes image features continuously in the spherical views of the nodes over time, was developed. The updating mechanism was inspired by the basic theory about human memory, which represents the memory using a multi-store model (Atkinson and Shiffrin, 1968).

Different experiments were conducted to test the proposed system, and were carried out in real changing environments over periods of up to nine weeks. During these periods, the adaptability of the map was tested using the similarity between the views stored in the nodes and those captured by the robot when revisiting the nodes. The results showed that using the proposed updating mechanism increased the similarity between the observed image features and the stored features in the map, which otherwise would have dropped over time if the stored view representations of the nodes remained static. The consistency of the map was also tested during a long-term navigation task. While the robot was updating the nodes daily, these nodes were used to estimate the heading direction required for the robot to navigate. The results showed that the robot was able to maintain consistent navigation performance over time and that the map did not degrade.

7.2 Meeting the Requirements

In the introduction to this thesis it was argued that any solution to the problem of long-term mapping in non-stationary environments should meet a number of requirements; here, we go back to those requirements to explain how our proposed system has fulfilled them:

- **Accuracy.** A recursive Unscented Kalman Filter (UKF) attached to every image feature on the spherical views of the map maintained the accuracy of

estimating their position over time, as shown in the experiment presented in Section 6.2, which in turn maintains the accuracy of estimating the robot's heading required for navigation. The results showed that the overall error in estimating the heading was around 4^o, which was shown to be sufficiently accurate for robust navigation using the map.

- **Computational requirements.** The hybrid structure of the map, i.e. the division into sparse nodes where each node contains a local self-contained spherical view, means that the computational requirements for the system should add rather than multiply. Also, the continuous forgetting and adding of new features to the map means that the number of image features stored in each node should not increase over time.

- **Stability.** The stability of the system would be compromised if image features generated from the views of one node became part of another node due to localisation failures. This case is avoided in the system due to the requirement that an image feature has to be seen many times in the STM store of a node before it can become part of the LTM store of that node. This means that occasional failures in localisation do not have a catastrophic effect on the map. This graceful degradation property of the system was shown in experiments presented in Section 5.6.1 and Section 6.3.

- **Delayed declaration of outliers.** This requirement refers to the fact that actual changes in the environment appear in the first instance as outliers, which can only be identified after more time has passed and more measurements have been made. In the system presented in this thesis, the rehearsal process in STM provides a solution to this requirement. The STM

keeps image features for a short period of time, so any features originated from moving objects or people or features from temporary occlusions can be removed.

- **Real-time versus off-line processing.** During long-term operations, the proposed system allowed the robot to perform localisation and visual navigation tasks under near real-time constraints, while the map updating process was performed off-line when the robot was considered inactive.

7.3 Limitations

Building a single system to cover every possible condition that a robot may encounter is far from achievable in one thesis. Therefore, there were always going to be limitations and areas which the system would not cover, which are outlined as follows along with a discussion on possible solutions.

This work did not address the problem of topological changes. It is assumed that the initial topological map built by the robot following its first tour in the environment is fixed. However, the topology of the environment may still change. Examples of such cases include the re-arrangement of office dividers on a company floor or the furniture in a living room, and in extreme cases removing or adding walls between two rooms.

One of the possible solutions for such cases would be to use the proposed memory model to adapt the topology of the map during long-term operation. When a new node is created, the robot can keep it in an STM store for a short period of time until it is confident that this node is ready to become part of the map. The same goes for the links between the nodes. If the robot stops being able

to use a link to travel between two nodes, the link would be removed from LTM. In this way, the same memory structure can be used for both the appearance and the topology of the environment. However, representing and reasoning about such changes in the environment is by no means a simple process. One can find a large body of literature which addresses this issue in the field of cognitive robotics (Hanks and McDermott, 1994; McDermott, 1982).

The system introduced in this thesis assumed that the environments where a robot operates are rich with structures that produce blob- and corner-like image features. However, a robot can face situations where the environment is poor in such features, e.g. a corridor with plain walls. In these situations the robot may not be able to localise itself or use the map for navigation due to the low number of image features. One possible solution to tackle this limitation is the use of multiple types of image features that complement each other. A strong candidate to compensate for the lack of blob- and corner-like features in parts of the robot's environment would be vertical line features, which are dominant in indoor environments due to their common rectilinear design. When using an omnidirectional vision system for a mobile robot working on a flat plane, these vertical line features appear as radial lines on the image plane, which makes them relatively easy to detect and track. Similar to blob- and corner-like features, vertical lines can be described by a high-dimensional vector extracted from the local surrounding regions around each line (Scaramuzza et al., 2009). These descriptors provide reliable matching results. Vertical lines can be stored in STM and LTM and used through multi-view geometry for visual navigation alongside SURF features. In situations when even vertical lines become scarce, the robot can switch to navigate using path integration (dead reckoning) or wall-following

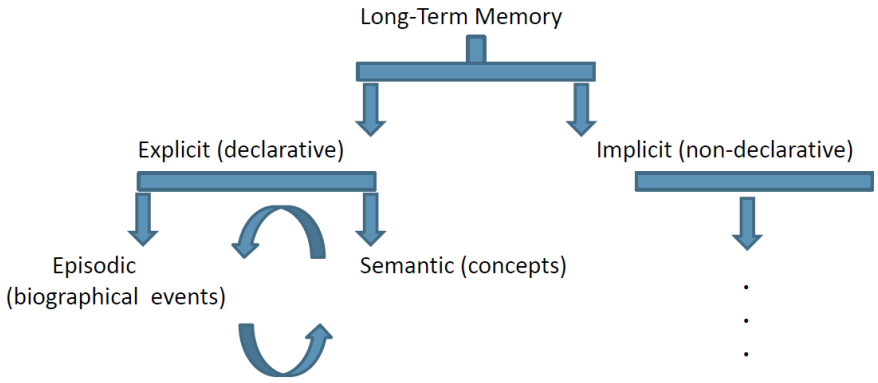

Figure 7.1: Long-term memory is divided into declarative and procedural memory, the former of which is divided further into semantic and episodic memory. Episodic memory provides the capacity to remember specific events, while semantic memory stores accumulated knowledge of the world. Forgetting plays an important role in maintaining a compact representation of the world for subsequent reasoning. Generalisation is believed to be one of the more important processes involved in improving the efficiency, scalability and adaptability of cognitive systems operating in dynamic environments (Baddeley et al., 2009).

behaviours to pass through the featureless parts of the environment.

7.4 Future Work

Memory is a process through which we can store and retrieve pieces of information. In this work, image features, which can be considered to represent these pieces of information, are used to recognise a place in a map by finding corresponding matches between features in LTM and the current view. The reasonable next step to extend the proposed system is to use these features to identify individual objects. The fact that they are generated from real objects in the environment means that when a subset of features appears in the view of the robot, the rest of the features from the same object are present in the environment, even though

they may not all be visible at the same time from the robot's point of view. The robot could then group the features which belong to the same objects using an object recognition system or an incremental learning process, which would enable it to create a so-called "semantic level" for the map and subsequently make any interaction between the user and the robot much more human-friendly by allowing them to communicate about the same objects.

In order to add a semantic level to the map, LTM can be investigated in more detail and take inspiration from the theory about its structure. Fig. 7.1 shows LTM in more detail, where it is divided into declarative and procedural memory. Under declarative memory, two types of memory can be found, namely semantic and episodic, the latter of which provides the capacity to remember specific events, while semantic memory stores accumulated knowledge of the world, e.g. some generalised representation of different episodes experienced (Baddeley et al., 2009).

Semantic mapping is by no means a simple problem. Various authors have proposed richer world models which incorporate semantic information, e.g. based on the identification of objects. Nuechter et al. (2003), for instance, developed an approach for labelling regions such as walls, floors, ceilings and doors in 3D range scans by extracting planes and then using prior knowledge to categorise the planes. Rottmann et al. (2005) developed a supervised learning approach for labelling different indoor locations such as offices, kitchens and corridors by training a classifier based on features extracted from vision and laser range data. Other works have investigated hierarchical maps where indoor spaces such as rooms can be further decomposed according to their functions and the objects contained therein (Galindo et al., 2005; Vasudevan and Siegwart, 2008; Zender

et al., 2008).

Common to such approaches is that the semantic level of information should help to improve the robustness of robotic mapping; for example, if a living room is decomposed into a space containing chairs and sofas, it can still be recognised when the chairs and sofas have been moved. However, adding semantic information to maps is not a general solution to the long-term mapping problem, since the functions of rooms, the location of objects and the structure of the environment may themselves change with time. Therefore, memory will be an essential component of service robots of the future, which will enable them to gradually adapt their semantic maps over time as the world around them changes.

References

T. Akihiko and I. Atsushi. Multiple View Geometry for Spherical Cameras. Technical report, Institute of Electronics, Information and Communication Engineers (IEIC), 2005.

J. Andrade-Cetto and A. Sanfeliu. Concurrent map building and localization in indoor dynamic environments. *International Journal of Pattern Recognition and Artificial Intelligence*, 16(3):361–374, 2002.

H. Andreasson, T. Duckett, and A. Lilienthal. A minimalistic approach to appearance based visual SLAM. *IEEE Transactions on Robotics*, 24(5):991–1001, 2008.

D. Anguelov, R. Biswas, D. Koller, B. Limketkai, S. Sanner, and S. Thrun. Learning Hierachical Objects Maps of Non-Stationary Environments with Mobile Robots. In *Proc. Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 10–17, Edmonton, Canada, August 2002.

R. Atkinson and R. Shiffrin. Human memory: A proposed system and its control processes. *The psychology of learning and motivation: Advances in research and theory*, 2:89–195, 1968.

A. Baddeley, M. W. Eysenck, and M. C. Anderson. *Memory*. Psychology Press, 2009.

S. Baker and S. K Nayar. A theory of single-viewpoint catadioptric image formation. *International Journal of Computer Vision*, 35(2):175–196, 1999.

H. Bay, T. Tuytelaars, and L. Van Gool. SURF: Speeded Up Robust Features. *Computer Vision and Image Understanding*, 110:346–359, 2008.

P. Beeson, J. Modayil, and B. Kuipers. Factoring the mapping problem: Mobile robot map-building in the hybrid spatial semantic hierarchy. *The International Journal of Robotics Research*, 29(4):428, 2010. ISSN 0278-3649.

J. Beis and D. Lowe. Indexing without invariants in 3D object recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(10):1000–1015, 1999.

C. Bibby and I. Reid. Simultaneous Localisation and Mapping in Dynamic Environments (SLAMIDE) with Reversible Data Association. In *Proc. Robotics: Science and Systems*, Atlanta, GA, USA, June 2007.

P. Biber and T. Duckett. Dynamic Maps for Long-Term Operation of Mobile Service Robots. In *Proc. Robotics: Science and Systems*, pages 17–24, Cambridge, MA, USA, June 8-11 2005.

G. Blanc, Y. Mezouar, and P. Martinet. Indoor navigation of a wheeled mobile robot along visual routes. In *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, pages 3354–3359, Orlando, Florida, USA, May 15-19 2006.

J. Blanco, J. Fernandez-Madrigal, and J. Gonzalez. Toward a Unified Bayesian Approach to Hybrid Metric-Topological SLAM. *IEEE Transactions on Robotics*, 24:259–270, 2008.

O. Booij, B. Terwijn, Z. Zivkovic, and B. Krose. Navigation using an appearance based topological map. In *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, pages 3927–3932, Roma, Italy, April 10-14 2007.

O. Booij, Z. Zivkovic, and B. Krose. Sampling in image space for vision based SLAM. In *Proc. The Inside Data Association Workshop during the Robotics: Science and Systems Conference (RSS)*, ETH Zurich, Switzerland, June 25-28 2008.

M. Bosse, P. Newman, J. Leonard, and S. Teller. Slam in large-scale cyclic environments using the atlas framework. *The International Journal of Robotics Research*, 23(12):1113–1139, 2004.

P. Buschka and A. Saffiotti. Some Notes on the Use of Hybrid Maps for Mobile Robots. In *Proc. International Conference on Intelligent Autonomous Systems (IAS)*, pages 547–556, Amsterdam, NL., 2004.

BA Cartwright and T.S. Collett. Landmark learning in bees. *Journal of Comparative Physiology A: Neuroethology, Sensory, Neural, and Behavioral Physiology*, 151(4):521–543, 1983.

H. Choset and K. Nagatani. Topological simultaneous localization and mapping (SLAM): toward exact localization without explicit localization. *IEEE Transactions on Robotics and Automation*, 17(2):125–137, 2001. ISSN 1042-296X.

H. Chris and S. Mike. A combined corner and edge detector. In *Proc. Alvey Vision Conference, (AVC)*, pages 147–152, Manchester, UK., 1988.

L. Clemente, A. Davison, I. Reid, J. Neira, and J. Tardos. Mapping Large Loops with a Single Hand-Held Camera. In *Proc. Robotics: Science and Systems*, Philadelphia, Pennsylvania, USA, August 16-19 2007.

N. Cristianini and J. Shawe-Taylor. *An introduction to support Vector Machines: and other kernel-based learning methods*. Cambridge university press, 2004. ISBN 0521780195.

M. Cummins and P. Newman. FAB-MAP: probabilistic localization and mapping in the space of appearance. *International Journal of Robotics Research*, 27(6): 647–665, 2008. ISSN 0278-3649.

K. Dannilidis and H. Nagel. The coupling of rotation and translation in motion estimation of planar surfaces. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 188–193, June 15-17 1993.

A. Davison, I. Reid, N. Molton, and O. Stasse. MonoSLAM: real-time single camera SLAM. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(6):1052–1067, 2007. ISSN 0162-8828.

T. Deselaers, D. Keysers, and H. Ney. Features for image retrieval: an experimental comparison. *Information Retrieval*, 11(2):77–107, 2008.

J. Dong, S. Wijesoma, and A. Shacklock. Extended rao-blackwellised genetic algorithmic filter SLAM in dynamic environment with raw sensor measurement. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1473–1478, San Diego, USA, Oct. 29 - Nov. 2 2007.

A. Doucet, N. de Freitas, K. Murphy, and S. Russell. Rao-Blackwellised particle filtering for dynamic Bayesian networks. In *Proc. Conference on Uncertainty in Artificial Intelligence*, pages 176–183, Stanford, CA, USA, June 30 - July 3 2000.

E. Eade and T. Drummond. Monocular SLAM as a Graph of Coalesced Observations. In *Proc. IEEE International Conference on Computer Vision (ICCV)*, pages 1–8, Rio de Janeiro, Brazil, October 14-20 2007.

A. Elfes. Using occupancy grids for mobile robot perception and navigation. *Computer*, 22(6):46–57, 1989. ISSN 0018-9162.

A. Eliazar and R. Parr. DP-SLAM: fast, robust simultaneous localization and mapping without predetermined landmarks. In *Proc. International Joint Conference on Artificial Intelligence*, volume 18, pages 1135–1142, Acapulco, Mexico, August 9-15 2003.

P. Elinas, R. Sim, and J. J. Little. sigmaSLAM: Stereo vision SLAM using the Rao-Blackwellised particle filter and a novel mixture proposal distribution. In *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, pages 1564–1570, Orlando, Florida, May 15-19 2006.

A. Ess, B. Leibe, K. Schindler, and L. van Gool. A Mobile Vision System for Robust Multi-Person Tracking. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, Anchorage, USA, June 2008.

O. Faugeras. *Three-Dimensional Computer Vision: A Geometric Viewpoint*. the MIT Press, 1993.

M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24:381–395, 1981.

D. Fox. *Markov Localization: A Probabilistic Framework for Mobile Robot Localization and Navigation.* PhD thesis, Institute of Computer Science III. University of Bonn, Bonn, Germany, 1998.

O. Frank, R. Katz, C. L Tisse, and H. Durrant-Whyte. Camera calibration for miniature, low-cost, wide-angle imaging systems. In *British Machine Vision Conference*, University of Warwick, UK, September 10-13 2007.

F. Fraundorfer, C. Engels, and D. Nistr. Topological mapping, localization and navigation using image collections. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3872–3877, San Diego, USA, Oct. 29 - Nov. 2 2007.

U. Frese and L. Schroder. Closing a million-landmarks loop. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5032–5039, Beijing, China, October 9 - 15 2006. ISBN 1424402581.

C. Galindo, A. Saffiotti, S. Coradeschi, P. Buschka, J.A. Fernández-Madrigal, and J. González. Multi-Hierarchical Semantic Maps for Mobile Robotics. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3492–3497, Edmonton, CA, August 2 - 6 2005.

B. Gates. A Robot in Every Home. *Scientific American*, pages 58–65, January 2007.

C. Geyer and K. Daniilidis. A unifying theory for central panoramic systems and practical implications. In *Proc. European Conference on Computer Vision (ECCV)*, pages 445–461, Dublin, Ireland, June 26 - July 1 2000.

Y. Girdhar and G. Dudek. ONSUM: a system for generating online navigation summaries. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 746–751, Taipei, Taiwan, October 18 - 22 2010.

T. Goedemé and L. Van Gool. Vision-only mobile robot navigation with topological maps. *Robot Vision: New Research*, pages 15–48, 2009.

T. Goedemé, M. Nuttin, T. Tuytelaars, and L. Van Gool. Omnidirectional Vision Based Topological Navigation. *International Journal of Computer Vision*, 74 (3):219–236, 2007.

N. Gordon, D. Salmond, and A. Smith. Novel approach to nonlinear/non-Gaussian Bayesian state estimation. *Radar and Signal Processing, IEE Proceedings F*, 140:107–113, April 2002.

G. Grisetti, C. Stachniss, and W. Burgard. Improved techniques for grid mapping with rao-blackwellized particle filters. *IEEE Transactions on Robotics*, 23(1): 34–46, 2007a.

G. Grisetti, C. Stachniss, S. Grzonka, and W. Burgard. A tree parameterization for efficiently computing maximum likelihood maps using gradient descent. In *Proc. Robotics: Science and Systems*, Atlanta, GA, USA, June 27-30 2007b.

G. Grisetti, C. Stachniss, and W. Burgard. Nonlinear constraint network optimization for efficient map learning. *IEEE Transactions on Intelligent Transportation Systems*, 10(3):428–439, 2009. ISSN 1524-9050.

H. Gross, A. Koenig, C. Schroeter, and H. Boehme. Omnivision-based probabilistic self-localization for a mobile shopping assistant continued. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1505–1511, Las Vegas, USA, October 27 - 31 2003.

S. Hanks and D. McDermott. Modeling a dynamic and uncertain world i:: Symbolic and probabilistic reasoning about change. *Artificial Intelligence*, 66(1): 1–55, 1994.

R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2 edition, March 2004. ISBN 0521540518.

S. Hochdorfer and C. Schlegel. Towards a robust visual SLAM approach: Addressing the challenge of life-long operation. In *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–6, Kope, Japan, May 12-17 2009.

B. K.P Horn. Relative orientation. *International Journal of Computer Vision*, 4 (1):5978, 1990.

S. Huang, Z. Wang, and G. Dissanayake. Mapping large scale environments using relative position information among landmarks. In *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, pages 2297–2302, Orlando, Florida, US, May 15-19 2006.

S. Huffel and J. Vandewalle. *The Total Least Squares Problem: Computational Aspects and Analysis*. Society for Industrial Mathematics, 1991.

V. Ila, J. M Porta, and J. Andrade-Cetto. Information-based compact pose SLAM. *IEEE Transactions on Robotics*, 26(1):78–93, 2010.

H. Ishiguro. Development of low-cost compact omnidirectional vision sensors and their applications. In *Proc. International Conference on Information systems, analysis and synthesis (ISAS)*, pages 433–439, 1998.

L. Itti and P. Baldi. Bayesian surprise attracts human attention. *Vision research*, 49(10):1295–1306, 2009. ISSN 0042-6989.

M. Jogan and A. Leonardis. Robust localization using an omnidirectional appearance-based subspace model of environment. *Robotics and Autonomous Systems*, 45(1):51–72, 2003. ISSN 0921-8890.

S. J Julier and J. K Uhlmann. A new extension of the Kalman filter to nonlinear systems. In *the International Symposium on Aerospace/Defense Sensing, Simulation and Controls*, volume 3, page 26, 1997.

M. Kaess, A. Ranganathan, and F. Dellaert. iSAM: incremental smoothing and mapping. *IEEE Transactions on Robotics*, 24(6):1365–1378, 2008. ISSN 1552-3098.

S. Kang. Catadioptric Self-Calibration. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 201–207, Kauai, HI, USA, December 8-14 2002.

S. Kang and R. Szeliski. 3-D scene data recovery using omnidirectional multibaseline stereo. *International Journal of Computer Vision*, 25:167–183, 1997.

K. Konolige and M. Agrawal. Frameslam: From bundle adjustment to real-time visual mapping. *IEEE Transactions on Robotics*, 24(5):1066–1077, 2008. ISSN 1552-3098.

K. Konolige and J. Bowman. Towards lifelong visual maps. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1156–1163, St. Louis, USA, October 11-15 2009.

K. Konolige, J. Bowman, J. D. Chen, P. Mihelich, M. Calonder, V. Lepetit, and P. Fua. View-Based Maps. *International Journal of Robotics Research*, 29(8): 941, 2010. ISSN 0278-3649.

J. Košecká, F. Li, and X. Yang. Global localization and relative positioning based on scale-invariant keypoints. *Robotics and Autonomous Systems*, 52: 27–38, 2005.

B. Kuipers. The spatial semantic hierarchy. *Artificial Intelligence*, 119:191–233, 2000.

B. Kuipers and Y. T. Byun. A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations. *Robotics and autonomous systems*, 8:47–63, 1991.

B. Kuipers, J. Modayil, P. Beeson, M. MacMahon, and F. Savelli. Local metrical and global topological maps in the hybrid spatial semantic hierarchy. In *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, volume 5, pages 4845–4851, New Orleans, USA, April 26 - May 1 2004. ISBN 0780382323.

F. Labrosse. The visual compass: Performance and limitations of an appearance-based method. *Journal of Field Robotics*, 23(10):913–941, 2006.

F. Labrosse. Short and long-range visual navigation using warped panoramic images. *Robotics and Autonomous Systems*, 55(9):675–684, 2007.

J. Li and N. M Allinson. A comprehensive review of current local features for computer vision. *Neurocomputing*, 71(10-12):1771–1787, 2008. ISSN 0925-2312.

C. Lin, K. Liu, and M. Chen. Dual clustering: integrating data clustering over optimization and constraint domains. *IEEE Transactions on Knowledge and Data Engineering*, 17(5):628–637, 2005. ISSN 1041-4347.

H. Longuet-Higgins. A computer algorithm for reconstructing a scene from two projections. *Nature*, 293:133–135, 1981.

D. Lowe. Object Recognition from Local Scale-Invariant Features. In *Proc. IEEE International Conference on Computer Vision (ICCV)*, pages 1150–1157, Kerkyra, Greece, September 20-27 1999.

F. Lu and E. Milios. Globally consistent range scan alignment for environment mapping. *Autonomous Robots*, 4(4):333–349, 1997.

J. Luo, A. Pronobis, B. Caputo, and P. Jensfelt. Incremental learning for place recognition in dynamic environments. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 721–728, San Diego, USA, Oct. 29 - Nov. 2 2007.

K. Lynch. *The Image of the City*. The MIT Press, June 1960. ISBN 0262620014.

M. J Mataric. Integration of representation into goal-driven behavior-based robots. *IEEE Transactions on Robotics and Automation*, 8(3):304–312, 1992. ISSN 1042-296X.

J. Matas, O. Chum, M. Urban, and T. Pajdla. Robust wide-baseline stereo from maximally stable extremal regions. *Image and Vision Computing*, 22(10):761–767, 2004.

Y. Matsumoto, K. Ikeda, M. Inaba, and H. Inoue. Visual navigation using omnidirectional view sequence. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, volume 1, pages 317–322, Lausanne, Switzerland, Sep. 30 - Oct. 4 2002. ISBN 0780351843.

N. Mavridis and M. Petychakis. Human-like memory systems for interactive robots: Desiderata and two case studies utilizing grounded situation models and online social networking. In *Symposium on Artificial Intelligence and Simulation of Behaviour (AISB)*, Leicester, UK, 2010.

D. McDermott. A temporal logic for reasoning about processes and plans. *Cognitive science*, 6(2):101–155, 1982.

E. Menegatti, M. Zoccarato, E. Pagello, and H. Ishiguro. Image-based monte carlo localisation with omnidirectional images. *Robotics and Autonomous Systems*, 48(1):17–30, 2004.

D. Migliore, R. Rigamonti, D. Marzorati, M. Matteucci, and D. G Sorrenti. Use a single camera for simultaneous localization and mapping with mobile object tracking in dynamic environments. In *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, pages 27–32, Kobe, Japan, May 12-17 2009.

K. Mikolajczyk and C. Schmid. Indexing based on scale invariant interest points. In *Proc. IEEE International Conference on Computer Vision (ICCV)*, volume 1, pages 525–531, Vancouver, Canada, July 7-14 2001. ISBN 0769511430.

M. Milford and G. Wyeth. Persistent navigation and mapping using a biologically inspired SLAM system. *The International Journal of Robotics Research*, 29(9): 1131–1153, 2010. ISSN 0278-3649.

N.C. Mitsou and C.S. Tzafestas. Temporal Occupancy Grid for mobile robot dynamic environment mapping. In *Proc. Mediterranean Conference on Conference on Control Automation*, pages 1–8, Kalamata, Greece, June 30 2007.

M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. FastSLAM: a factored solution to the simultaneous localization and mapping problem. In *Proc. National Conference on Artificial Intelligence (AAAI)*, pages 593–598, Alberta, Canada, July 28 August 1 2002.

A. Morris, D. Silver, D. Ferguson, and S. Thayer. Towards topological exploration of abandoned mines. In *Proc. IEEE International Conference on Robotics and*

Automation (ICRA), pages 2117–2123, Barcelona, Spain, April 18-22 2005. ISBN 078038914X.

A. Murillo, J. Guerrero, and C. Sagues. SURF features for efficient robot localization with omnidirectional images. In *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, pages 3901–3907, Roma, Italy, April 10-14 2007a. ISBN 1050-4729.

A.C. Murillo, C. Sagues, J.J. Guerrero, T. Goedemé, T. Tuytelaars, and L. Van Gool. From omnidirectional images to hierarchical localization. *Robotics and Autonomous Systems*, 55(5):372–382, 2007b.

S. Nayar and S. Baker. Catadioptric Image Formation. In *Proc. DARPA Image Understanding Workshop*, pages 1431–1437, 1997.

P. M. Newman. *On the structure and solution of the simultaneous localization and mapping problem*. PhD thesis, University of Sydney, Australia, 1999.

D. Nister and H. Stewenius. Scalable Recognition with a Vocabulary Tree. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 2161–2168, New York, NY, USA, June 17-22 2006.

D. Nister, O. Naroditsky, and J. Bergen. Visual odometry for ground vehicle applications. *Journal of Field Robotics*, 23:3–20, 2006.

A. Nuechter, H. Surmann, K. Lingemann, and J. Hertzberg. Semantic analysis of scanned 3d indoor environments. In *Proc. International Workshop on Vision, Modelling, and Visualization (VMV)*, pages 215–222, Munich, Germany, November 2003.

T. Ohno, A. Ohya, and S. Yuta. Autonomous navigation for mobile robots referring pre-recorded image sequence. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, volume 2, pages 672–679, Lausanne, Switzerland, Sep. 30 - Oct. 4 2002. ISBN 078033213X.

E. Olson, J. Leonard, and S. Teller. Fast iterative alignment of pose graphs with poor initial estimates. In *Proc. IEEE International Conference on Robotics*

and Automation (ICRA), pages 2262–2269, Orlando, Florida, USA, May 15-19 2006.

S. Ortega. *Towards visual localization, mapping and moving objects tracking by a mobile robot: a geometric and probabilistic approach*. PhD thesis, Institut National Polytechnique de Toulouse, 2007.

C. Peters. An Object-Based Memory for Supporting Attentive Virtual Agents. In *Symposium on Artificial Intelligence and Simulation of Behaviour (AISB)*, Leicester, UK, March 29- April 1 2010.

R.A. Peters, K.A. Hambuchen, K. Kawamura, and D.M. Wilkes. The Sensory Egosphere as a Short-Term Memory for Humanoids. In *Proc. of the IEEE/RAS International Conference on Humanoid Robots*, pages 451–459, Tokyo, Japan, 2001.

A. Ranganathan and F. Dellaert. Bayesian surprise and landmark detection. In *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, pages 2017–2023, Kobe, Japan, May 12-17 2009.

A. Ranganathan and F. Dellaert. Online probabilistic topological mapping. *The International Journal of Robotics Research*, 30(6):755–771, 2011. ISSN 0278-3649.

E. Rosten and T. Drummond. Machine learning for high-speed corner detection. In *Proc. European Conference on Computer Vision (ECCV)*, pages 430–443, Graz, Austria, May 7-13 2006.

A. Rottmann, O. Martínez Mozos, C. Stachniss, and W. Burgard. Place Classification of Indoor Environments with Mobile Robots using Boosting. In *Proc. National Conference on Artificial Intelligence (AAAI)*, volume 20, pages 1306–1311, Pittsburgh, PA, USA, July 9-13 2005.

R. Rubinstein and D. Kroese. *Simulation and the Monte Carlo method*, volume 707. Wiley-interscience, 2008.

D. Scaramuzza, A. Martinelli, and R. Siegwart. A Toolbox for Easily Calibrating Omnidirectional Cameras. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5695–5701, Beijing, China, October 9 - 15 2006.

D. Scaramuzza, R. Siegwart, and A. Martinelli. A robust descriptor for tracking vertical lines in omnidirectional images and its use in mobile robotics. *The International Journal of Robotics Research*, 28(2):149, 2009.

S. Se, D. Lowe, and J. Little. Vision-based mobile robot localization and mapping using scale-invariant features. In *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, volume 2, pages 2051–2058, Seoul, Korea, May 21-26 2001.

S. Se, D. Lowe, and J. Little. Mobile Robot Localization and Mapping with Uncertainty using Scale-Invariant Visual Landmarks. *The International Journal of Robotics Research*, 21(8):735, 2002.

M. Selekwa, E. Collins, and J. Combey. Multivalued Verus univalued Reactive Fuzzy Behavior Systems for Navigation Control of Autonomous Ground Vehicles. In *Proc. Florida Conference on the Recent Advances in Robotics (FCRAR)*, volume 20, 2004.

A. W. Siegel, S. H. White, and H. W. Reese. The Development of Spatial Representations of Large-Scale Environments. In *Advances in Child Development and Behavior*, volume 10, pages 9–55. Academic Press, New York, 1975. ISBN 0065-2407.

R. Sim, P. Elinas, M. Griffin, and J. J Little. Vision-based SLAM using the Rao-Blackwellised particle filter. In *IJCAI Workshop on Reasoning with Uncertainty in Robotics*, pages 9–16, Edinburgh, Scotland, UK, August 5 2005.

R. Smith, M. Self, and P. Cheeseman. A stochastic map for uncertain spatial relationships. In *Proc. International symposium on robotics research*, pages 467–474. MIT Press Cambridge, MA, USA, 1987.

C. Stachniss and W. Burgard. Mobile robot mapping and localization in non-static environments. In *Proc. National Conference on Artificial Intelligence (AAAI)*, volume 20, pages 1324–1329, Pittsburgh, Pennsylvania, USA, July 9-13 2005.

T. Svoboda and T. Pajdla. Epipolar geometry for central catadioptric cameras. *International Journal of Computer Vision*, 49:23–37, 2002.

J. D Tardos, J. Neira, P. M Newman, and J. J Leonard. Robust mapping and localization in indoor environments using sonar data. *The International Journal of Robotics Research*, 21(4):311, 2002.

S. Thrun and J. Leonard. Simultaneous Localization and Mapping. In Bruno Siciliano and Oussama Khatib, editors, *Springer Handbook of Robotics*, chapter 37, pages 871–889. Springer-Verlag New York Inc, New York, 2008.

S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. MIT Press, 2005.

N. Tomatis, I. Nourbakhsh, and R. Siegwart. Hybrid simultaneous localization and map building: a natural integration of topological and metric. *Robotics and Autonomous Systems*, 44(1):3–14, 2003.

E. Tulving. Episodic and semantic memory. *Organization of memory*, pages 381–402, 1972.

J. K Uhlmann. Simultaneous map building and localization for real time applications. *transfer thesis, University of Oxford, Oxford, UK*, 1994.

I. Ulrich and I. Nourbakhsh. Appearance-based place recognition for topological localization. In *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, pages 1023–1029, San Francisco, CA, USA, April 24-28 2000.

C. Valgren, A. Lilienthal, and T. Duckett. Incremental topological mapping using omnidirectional vision. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3441–3447, Beijing, China, October 9 - 15 2006.

P. Vargas, A. Freitas, M. Lim, S. Enz, W. Ching Ho, and R. Aylett. Forgetting and Generalisation in Memory Modelling for Robot Companions: a Data Mining Approach. In *Symposium on Artificial Intelligence and Simulation of Behaviour (AISB)*, Leicester, UK, March 29- April 1 2010.

S. Vasudevan and R. Siegwart. Bayesian space conceptualization and place classification for semantic maps in mobile robotics. *Robotics and Autonomous Systems*, 56(6):522–537, June 2008.

P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 500–511, Kauai, HI, USA, December 8-14 2001.

N. Vlassis, B. Terwijn, and B. Krose. Auxiliary particle filter robot localization from high-dimensional sensor observations. In *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, pages 7–12, Washington, D.C., USA, May 11-15 2002.

A. N. Walcott. *Long-term Robot Mapping in Dynamic Environments*. PhD thesis, Massachusetts Institute of Technology, Department of Electrical Engineering and Computer Science., June 2011.

C. C Wang, C. Thorpe, S. Thrun, M. Hebert, and H. Durrant-Whyte. Simultaneous localization, mapping and moving object tracking. *The International Journal of Robotics Research*, 26(9):889, 2007. ISSN 0278-3649.

C.-W. Wang, C. Thorpe, and S. Thrun. Online SLAM with detection and tracking of moving objects: Theory and results from a ground vehicle in crowded urban areas. In *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, pages 842–849, Taipei, Taiwan, 2003.

D. F Wolf and G. S Sukhatme. Mobile robot simultaneous localization and mapping in dynamic environments. *Autonomous Robots*, 19(1):53–65, 2005. ISSN 0929-5593.

B. Yamauchi and R. Beer. Spatial Learning for Navigation in Dynamic Environments. *IEEE Transactions on Systems, Man and Cybernetics*, 26(3):496–505, 1996.

K. Yamazawa, Y. Yagi, and M. Yachida. Omnidirectional imaging with hyperboloidal projection. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, volume 2, pages 1029–1034, Tokyo, Japan, July 26 - 30 1993.

X. Ying and Z. Hu. Catadioptric camera calibration using geometric invariants. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1260–1271, 2004. ISSN 0162-8828.

H. Zender, O. Mozos, P. Jensfelt, G. Kruijff, and W. Burgard. Conceptual Spatial Representations for Indoor Mobile Robots. *Robotics and Autonomous Systems*, 56(6):493–502, June 2008.

Z. Zhang. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1330–1334, 2002. ISSN 0162-8828.

J. Zhou, F. Bian, J. Guan, and M. Zhang. Fast Implementation of Dual Clustering Algorithm for Spatial Data Mining. In *Proc. International Conference on Fuzzy Systems and Knowledge Discovery*, volume 3, pages 568–572, Los Alamitos, CA, USA, 2007.

Z. Zivkovic and O. Booij. How did we built our hyperbolic mirror omnidirectional camera-practical issues and basic geometry. Technical Report IAS-UVA-05-04, University of Amsterdam, 2005.

Z. Zivkovic, O. Booij, and B. Kröse. From images to rooms. *Robotics and Autonomous Systems*, 55(5):411–418, 2007.